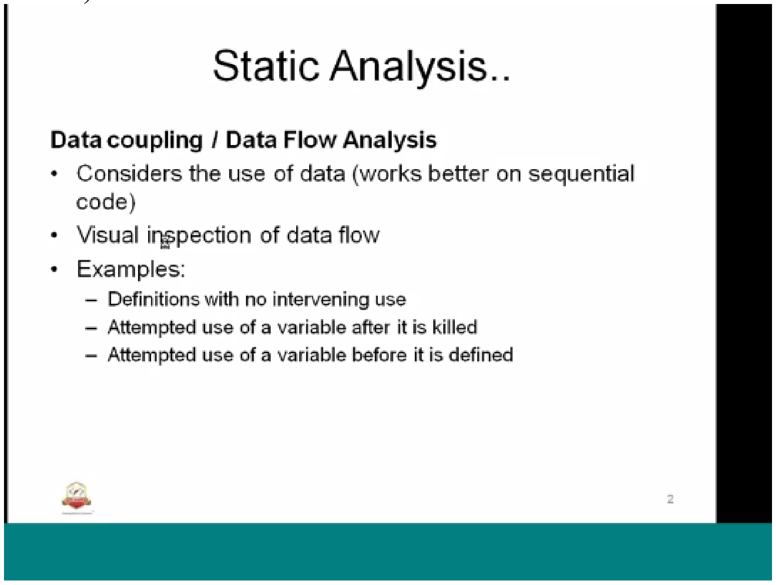
Hello and thanking you all for joining the next session of embedded software testing. We are continuing the unit three that is on static analysis reviews and matrix and this is lecture of the, that unit series. So today we study about the statics analysis aspects, before that we try to recap. So what we have to study in the earlier static analysis sessions.
(Refer Slide Time: 00:41)



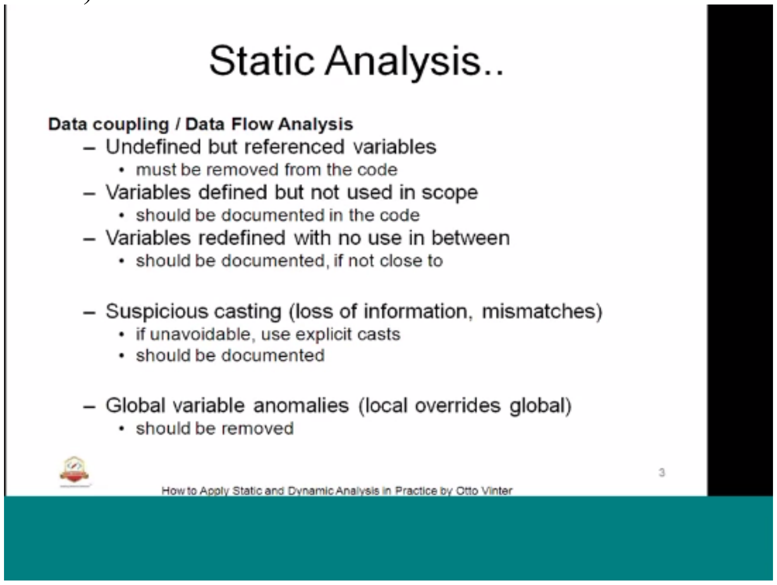
Static Analysis..

Data coupling / Data Flow Analysis

- Considers the use of data (works better on sequential code)
- Visual inspection of data flow
- Examples:
  - Definitions with no intervening use
  - Attempted use of a variable after it is killed
  - Attempted use of a variable before it is defined

2

So we continued about the static analysis control coupling and data coupling data flow analysis. We know that the program path and the worst case path. We will be explaining the terms of the program instructor and all that in the control and in terms of data coupling and data flow. We consider the use of the data or it is going to inter face between two different functionality and procedures or and reference variable.
(Refer Slide Time: 01:14)



Static Analysis..

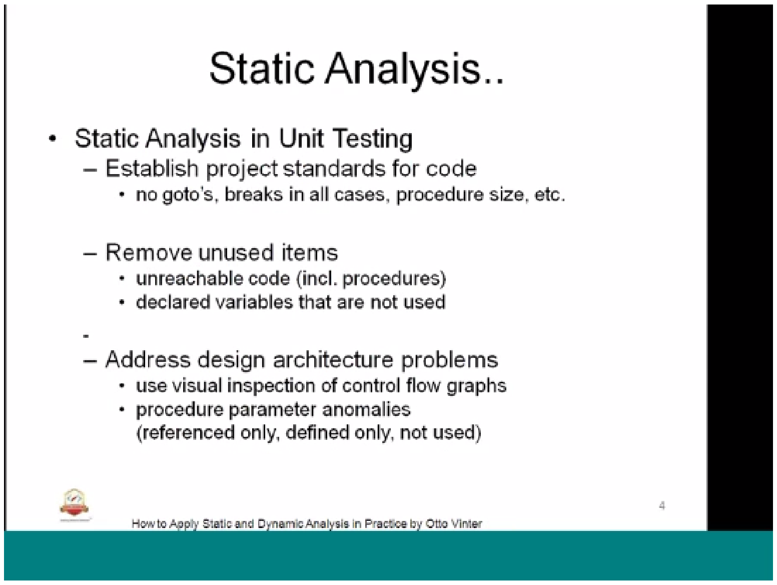
Data coupling / Data Flow Analysis

- Undefined but referenced variables
  - must be removed from the code
- Variables defined but not used in scope
  - should be documented in the code
- Variables redefined with no use in between
  - should be documented, if not close to
- Suspicious casting (loss of information, mismatches)
  - if unavoidable, use explicit casts
  - should be documented
- Global variable anomalies (local overrides global)
  - should be removed

3

How to Apply Static and Dynamic Analysis in Practice by Otto Vinter

Define how it is being type casted available variables and analysis whether same name are used local etc.
(Refer Slide Time: 01:23)



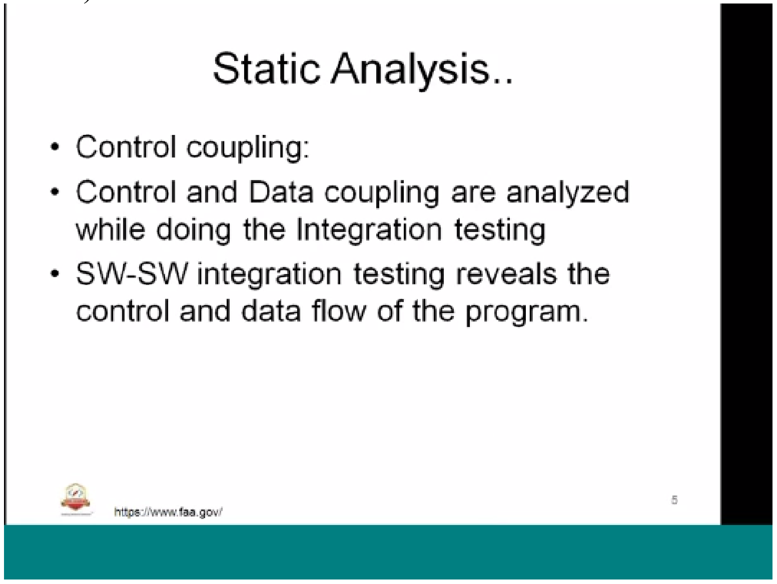
Static Analysis..

- Static Analysis in Unit Testing
  - Establish project standards for code
    - no goto's, breaks in all cases, procedure size, etc.
  - Remove unused items
    - unreachable code (incl. procedures)
    - declared variables that are not used
  -
  - Address design architecture problems
    - use visual inspection of control flow graphs
    - procedure parameter anomalies (referenced only, defined only, not used)

How to Apply Static and Dynamic Analysis in Practice by Otto Vinter

4

All this will be analysis statically with the help of the program so basically static analysis will be done during the in testing or during the integrations testing but the effaces during testing is on process standard code as per that is need to be analysis against and such as no go to breaks and all this, all this will be analysis to see that the correct program and the program it will be safety definitions of the unit for this being implemented.
Similarly unused items which are the unreachable codes unreachable variables will be have to be removed that also to be analysis during the static analysis similarly the architecture aspect of the program such as designs lower level or higher level designs also will be instructed of the control of the data flow so their we will see how the parameters are flow in between different function and the how the function flow.
(Refer Slide Time: 02:46)



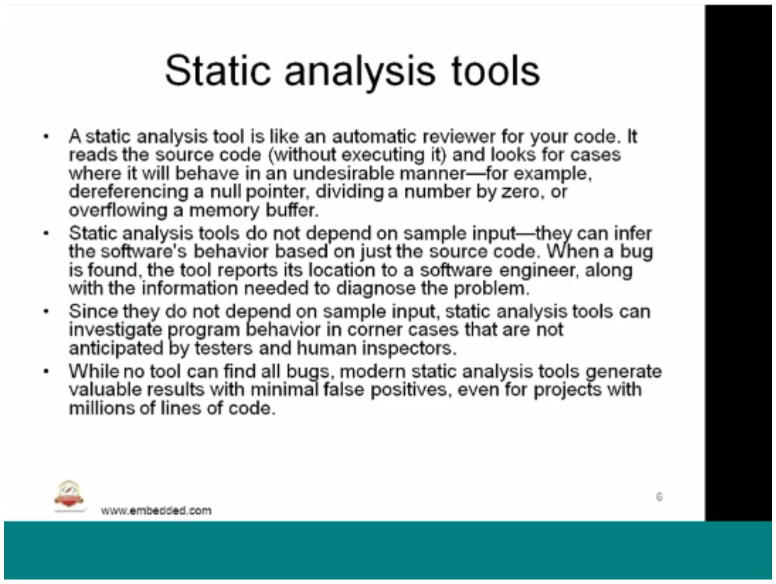
Static Analysis..

- Control coupling:
- Control and Data coupling are analyzed while doing the Integration testing
- SW-SW integration testing reveals the control and data flow of the program.

https://www.faa.gov/

5

So control coupling and data coupling can be designed
(Refer Slide Time: 02:48)



Static analysis tools

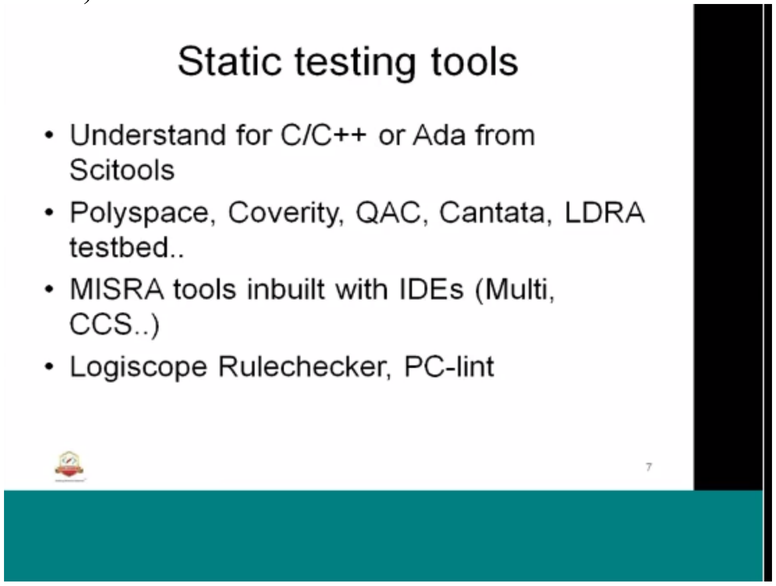
- A static analysis tool is like an automatic reviewer for your code. It reads the source code (without executing it) and looks for cases where it will behave in an undesirable manner—for example, dereferencing a null pointer, dividing a number by zero, or overflowing a memory buffer.
- Static analysis tools do not depend on sample input—they can infer the software's behavior based on just the source code. When a bug is found, the tool reports its location to a software engineer, along with the information needed to diagnose the problem.
- Since they do not depend on sample input, static analysis tools can investigate program behavior in corner cases that are not anticipated by testers and human inspectors.
- While no tool can find all bugs, modern static analysis tools generate valuable results with minimal false positives, even for projects with millions of lines of code.

www.embedded.com

6

in the integrations mostly the integrations could be software, software integrations or the hardware software integrations so next one, we had gone through some of the static analysis tools.

(Refer Slide Time: 03:08)



Static testing tools

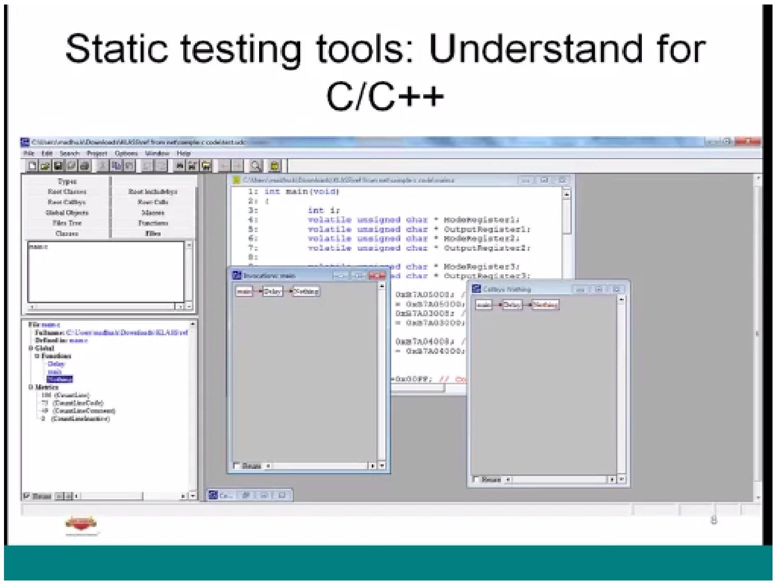
- Understand for C/C++ or Ada from Scitools
- Polyspace, Coverity, QAC, Cantata, LDRA testbed..
- MISRA tools inbuilt with IDEs (Multi, CCS..)
- Logiscope Rulechecker, PC-lint

7

Such a understand for C++ or Ada from sctitools, polyspace, coverity QAC, cantata, LDRA testbed all this can be used for static analysis basically they provided the quality and the usage of the variables the complete rate outline code the MacIver complexity what we have study and of course.

To the in builder tools such as MISRA tool will be part of the ID the integrated development itself so we can run through the tools such that it generate the report that what are the info type errors or what are the warning of the regulations that what we have in terms of MISRA rules similarly we have logicscope, rulechecker, PC-lint likewise also.

(Refer Slide Time: 04:06)



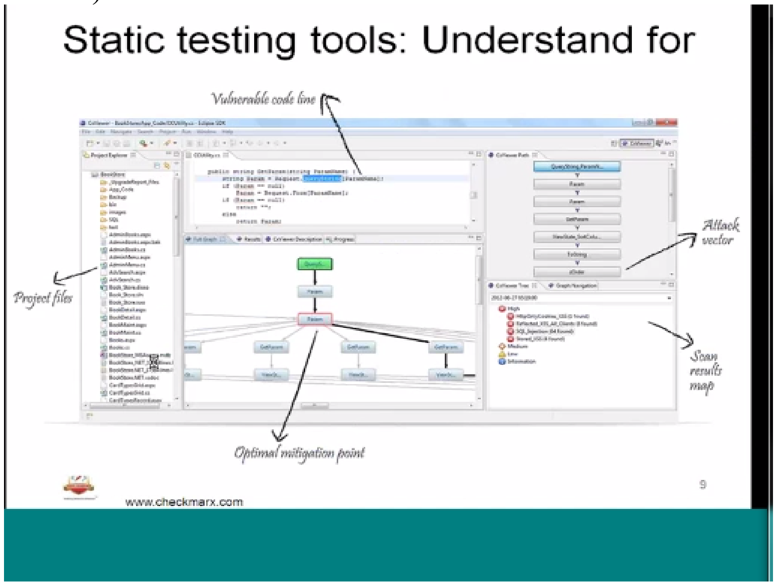
Static testing tools: Understand for C/C++

```
int main(void)
{
        int i;
        volatile unsigned char * ModeRegister1;
        volatile unsigned char * OutputRegister1;
        volatile unsigned char * ModeRegister2;
        volatile unsigned char * OutputRegister2;

        volatile unsigned char * ModeRegister3;
        volatile unsigned char * OutputRegister3;
```

We had gone through sample of that sort of understand for C/C++ here you can see the invocations so with the help of the invocations of different functions we can understand the control flow similarly the variable unused objects all this will be analysis with the help of this two code.

(Refer Sli8deTime: 04:18)

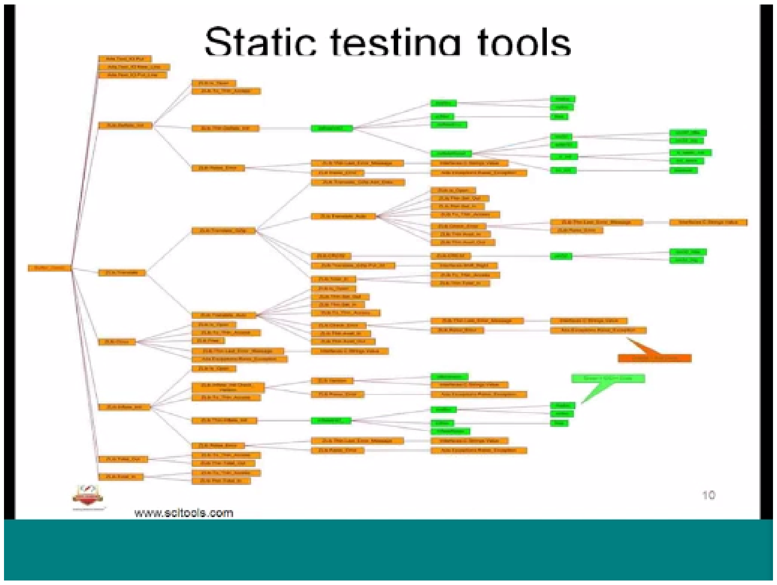


Static testing tools: Understand for

```
public string GetParam(string ParamName) {
    string Param = Request.QueryString[ParamName];
    if (Param == null)
        Param = Request.Form[ParamName];
    if (Param == null)
        return "";
    else
        return Param;
```

The other tools are check marks.

(Refer Slide Time: 04:23)

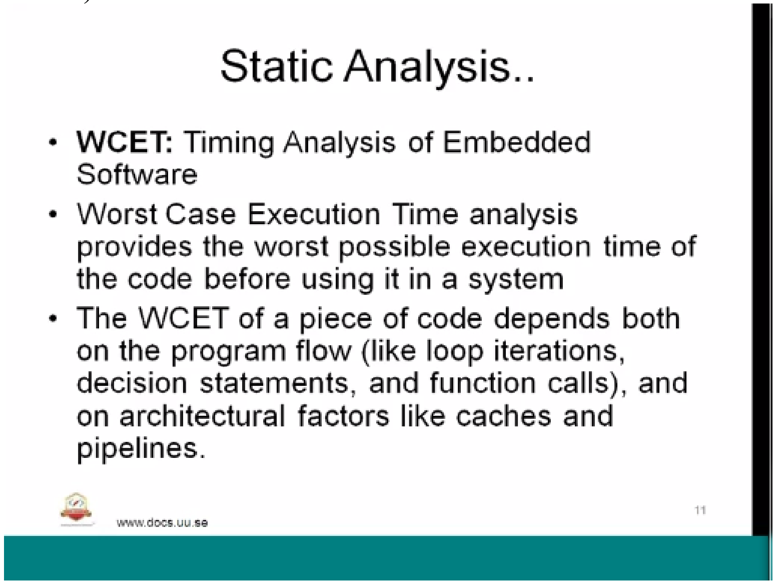

Static testing tools

www.scitools.com

10

So one sample report also we have seen in terms of how deep the control flow can go from left to the right most you can see the various procedure of the functionally project called so typical this is the tree call tree that we be produced during the static analysis.

(Refer Slide Time: 04:38)


Static Analysis..

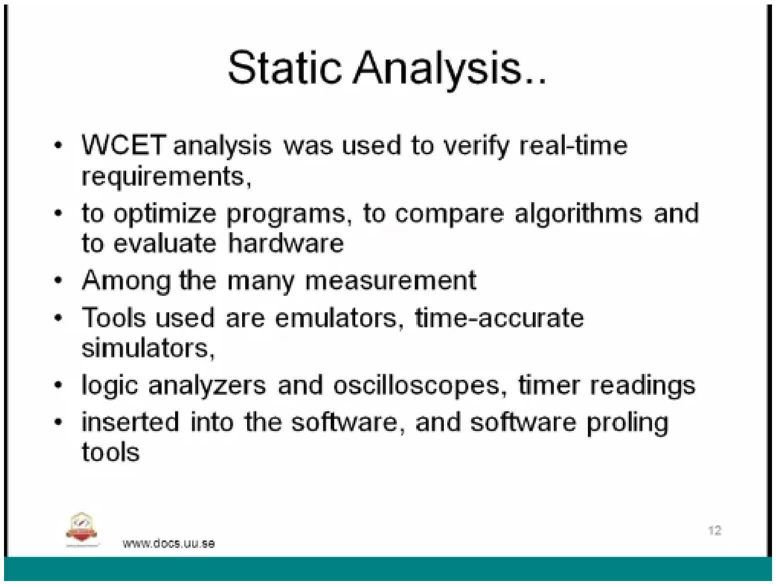
- **WCET:** Timing Analysis of Embedded Software
- Worst Case Execution Time analysis provides the worst possible execution time of the code before using it in a system
- The WCET of a piece of code depends both on the program flow (like loop iterations, decision statements, and function calls), and on architectural factors like caches and pipelines.

www.docs.uu.se

11

As part of the static analysis we also do the WCET and the VCET stands for vest case executions time and WCET stands for worst case execution time so basically it provides what is the worst possible executions time and vest possible executions path that system can take during it is executions.

(Refer Slide Time: 05:06)



Static Analysis..

- WCET analysis was used to verify real-time requirements,
- to optimize programs, to compare algorithms and to evaluate hardware
- Among the many measurement
- Tools used are emulators, time-accurate simulators,
- logic analyzers and oscilloscopes, timer readings
- inserted into the software, and software proling tools

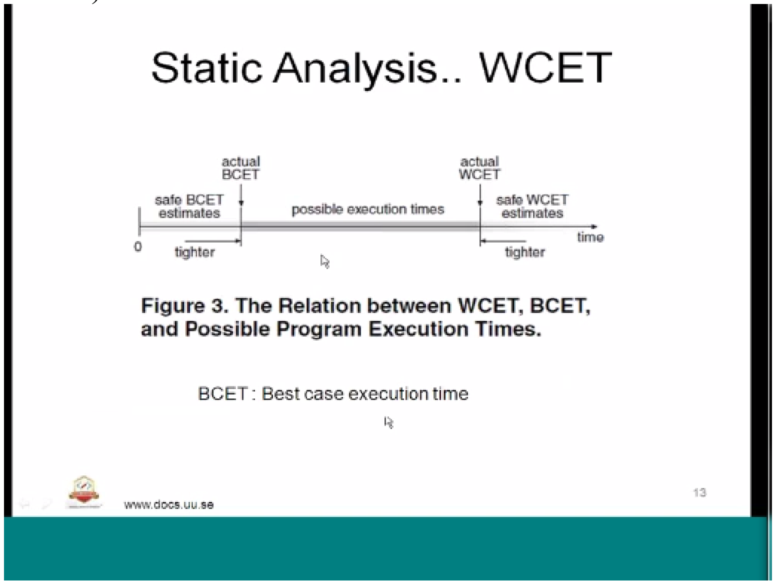
www.docs.uu.se 12

So that will be analysis against the stringent of the performance of the real time requirements of the implemented embedded system program so that also we will help in terms of how much we can optimize which functionality it can optimize in terms of comparing variables units such as algorithms any close-up control system all this will be measure so that is the important task of static analysis.

That the WCET control flow and data flow have to be measure so additionally for timing requirements, memory requirements and that we use logic analysis oscilloscope and any support tools such as emulators' time accurate also there is a other aspects for static analysis that can be done with the help of the test hook or the test code that will inflected into the software and there are software profiling tools.

Which will help in terms of analyzing the call tar strake memory strake analysis or stake over flow we will study to the today, all this will be done will the help of the interacted code so that is also the part of static analysis.

.(Refer Slide Time: 06:24)



Static Analysis.. WCET

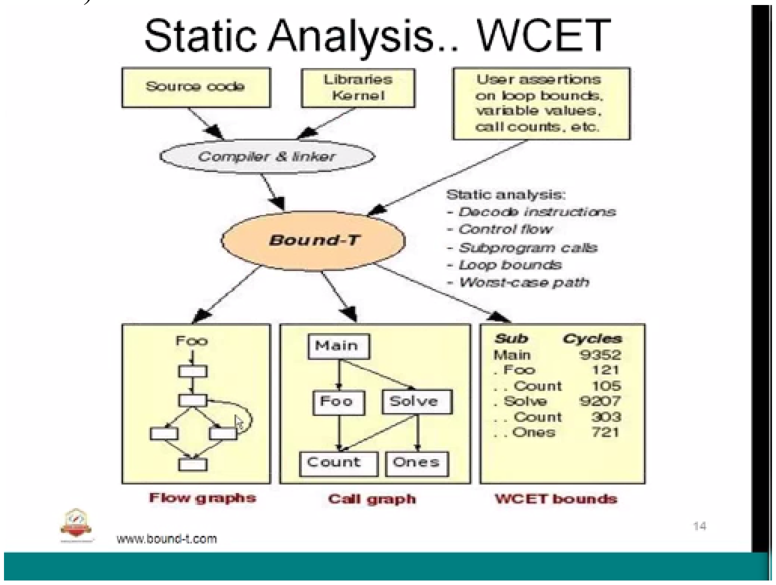
Figure 3. The Relation between WCET, BCET, and Possible Program Execution Times.

BCET : Best case execution time

www.docs.uu.se 13

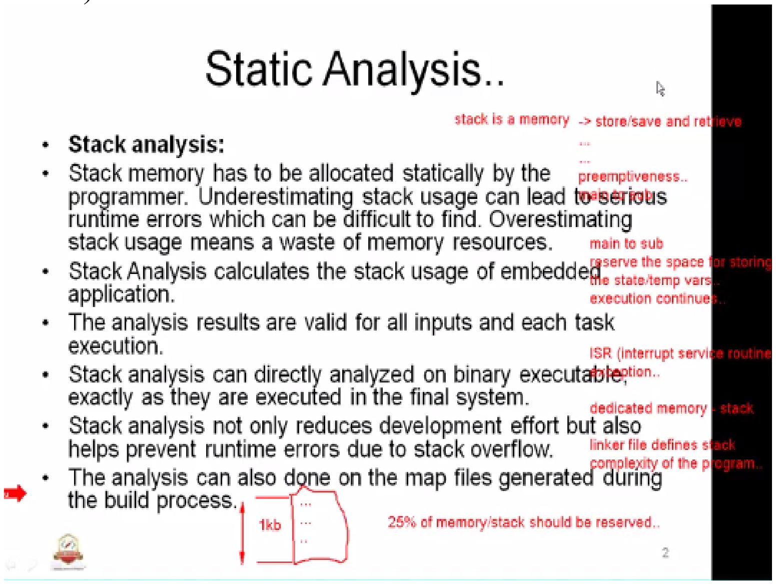
Here we can see examples of a static analysis the relationship between WCET, BCET and possible program executions times on the right hand side which terms the time and we will define the possible executions time at the center and it right an executions time of the safe executions times likewise.

(Refer Slide Time: 06:47)



So there is bound T analysis we call it so all the top bound will be done the pre analysis and the post analysis static analysis to be done on the functions that are implied with the program in terms of flow graph, call graph, and the WCET bounds it will produced. So coming to sessions static analysis and stack over flow and stack over-EASA guidance we will study about this okay.

(Refer Slide Time: 07:38)



Static Analysis..

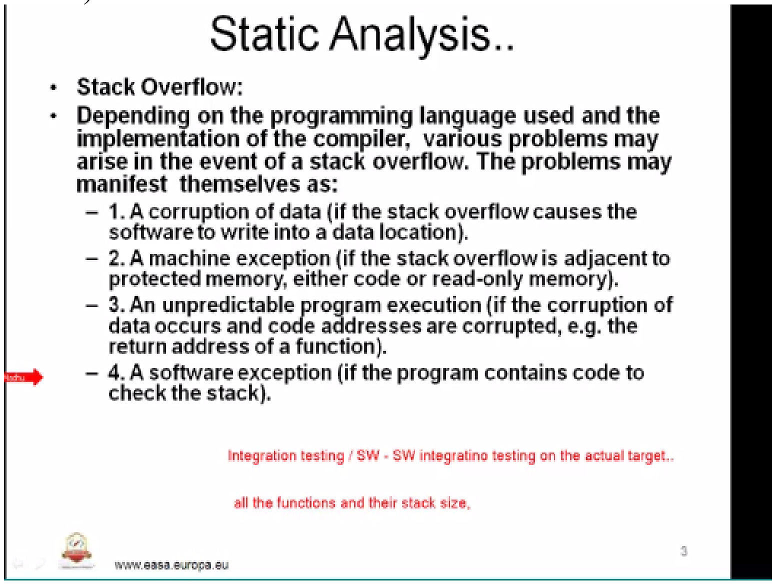
- **Stack analysis:**
- Stack memory has to be allocated statically by the programmer. Underestimating stack usage can lead to serious runtime errors which can be difficult to find. Overestimating stack usage means a waste of memory resources.
- Stack Analysis calculates the stack usage of embedded application.
- The analysis results are valid for all inputs and each task execution.
- Stack analysis can directly analyzed on binary executable, exactly as they are executed in the final system.
- Stack analysis not only reduces development effort but also helps prevent runtime errors due to stack overflow.
- The analysis can also done on the map files generated during the build process.

stack is a memory -> store/save and retrieve
...
...
preemptiveness..
main to sub
main to sub
reserve the space for storing the state/temp vars..
execution continues..
ISR (interrupt service routine
exception..
dedicated memory - stack
linker file defines stack
complexity of the program..
1kb
25% of memory/stack should be reserved..
2

So what is static analysis you know what is stack analysis which is memory part, part of the program which will be used dynamically during the execution of the program when there is permission or the entry and exist of the different functions of the producers the stack will be used for rather purpose of saving the state and the retrieving the state.

Stack is the memory part of the embedded program which will have store or save and retrieve that will be keep on doing over the execution of the embedded program such that there is no loses of data during preemptiveness basically it will help in preemativeness of a functions of the main functions are supporting from the main resources, main to sub in order to resource some of the variables during the executions this will be used main to sub.
And reserve the space for storing the state or temporary variables whatever it is depending on the state the where main revere gas used so it means to save the current and the flow of the variables that is used and once the sub routine call are done going back to the pervious routine all saved one will be retrieved and the execution will continue so similarly we have ISR interrupt service routine.
So what it does is involved the reserve ISR or it is also called has a exceptions so what will have up on this the ISR take the priority and before the ISR team working whatever that the permission happen on the routine or the producers it will save a state of that routine in terms of states variables or the program counter where it was then when the ISR is return that state will be retrieved back.
So for this there is a dedicated it is called stack so this stack has to be define during the link or a you know that compassion at the beginning is two aspects either result of linking we know that object files are reorganized along with the map of the complete embedded system program so what we do there is a link of file which defines the, all the memory basically but in particular stack which we are of interest now so this stack basically.
But it will be in terms of how much is the program is going to be is basically based on the complicity and the level of nesting and the sub routine called the ISR, all this matters so based on that the stack will define it could have one k to k whatever it is depending on the complicity of the program so the stack will be define and the tele point of time the stack need to be within the certain range that means so what will happen suppose.
Let we assume that stack and total size is, is a lower bound and it is upper bound and we have a say one kilo bites so what will happen is area will be used utilize by the program. and the stack will usually grow from the lower to upper side usually whenever there events like sub program sub routines substances that are executed or the prevention from the ISR this stack memory will be used in order to save the states.
And the variables in track to be stack and it will starts from lower to higher likewise till the last space so we have to make sure that while doing the linking we allocate the sufficient stack such that a tele point of stack it will not over flow that means the stack cannot go behind the allocate memory so that is what we have to ensure how this will be done there are different tools and techniques that are used but this will be part of the static analysis.
So that is what the stack of stack analysis so basically we need to see the static program how much it is the allocated and the and the stack user will be calculated based on the embedded applications how much it can be obtain okay so stack analysis. Stack memory is need to allocated statically and where the programmer are in set during the compliant time so it will be allocated under estimating stack can give to series run time errors.
Because we do know the behavior of the program because it is not able to save because already the stack memory is over flown and is not predictable so definitely we need to have buffer of this

stack a tele point of time during the program executions so it is difficult to find the stack issues over estimating the stack issues, more estimating stack issues to much memory we have and user, a very less space of stack, so typically memory will see suggest and define to have at least 52 and 75 percent of usage so that, there is a 25 percent of a stack has buffer,
So that is that standard of employee it is about 25 percent of a memory, stack should be resource this is what specify , we should not have the overestimate as well as, 4 and increase stack that is what it is very important, and stack analysis calculate the stack usage of embedded application , the analysis result are where valid for all the inputs and each task execution, definitely each execution use these stack, so that result will aware it and high light all this analysis, stack analysis can directly analyzed on binary executable is we know the binary executive is the 1, should be running, so the embedded software execution, exactly.
It is different from the final issues stack analysis not only reduces development effort but also, helps the prevent run time errors, definitely stack overflow is one of the typical common errors, that we have see, especially in the application field such as or whatever it is, but in the embedded applications where not cleared about, where the programs is going to give or where it is not cleared design.
How much build process is and he would have defined and limited memory, differently will give chance that so those who are occurs? Why? Because programmer knows about the program what he has developed? there are different things along with the program what he has developed such as library used which is not aware of how much it is working on there are certain assembly, it is occupied stack and there are certain support functions are, some third parties hooks are code which is not aware so those of the areas where you see, there is a stack over flow.
The channels of stack over flow is there so that is to be analysis that is what stack analysis primenerly the analysis has be done on the map files the generated during the bills, that is the one side bill after the linking we generate the along with the binary executing of the map file, the map file will clearly tell how much executing stages and all that, so based on that we can analysis that is about stack analysis.
(Refer Slide Time: 18:36)



Static Analysis..

- **Stack Overflow:**
- **Depending on the programming language used and the implementation of the compiler, various problems may arise in the event of a stack overflow. The problems may manifest themselves as:**
  - 1. A corruption of data (if the stack overflow causes the software to write into a data location).
  - 2. A machine exception (if the stack overflow is adjacent to protected memory, either code or read-only memory).
  - 3. An unpredictable program execution (if the corruption of data occurs and code addresses are corrupted, e.g. the return address of a function).
  - 4. A software exception (if the program contains code to check the stack).

Integration testing / SW - SW integratino testing on the actual target..

all the functions and their stack size,

www.easa.europa.eu

3

Next one so there is a stack over flow problem so what is that mean stack over flow we will study that depending on the programming language and the implementations of the complier various problems may arise in the event of stack over flow the problem may manifest themselves as that means what are the problem that we are going to come across in register over flow so what are those the corruptions of data.

If the stack over flow cases the software to write into a data locations definitely we are going to have unintentional right into data locations that can cause the corruption of the data machine exceptions because there is a stack over flow adjacent to protected memory and we are try to either or access the code which are read only memory defiantly there is a unwanted executions for this then an unpredictable program executions.

That means the execution cannot be control will be the correction of data occurs and code addresses are corrupted definitely the can behave level we do not know the return addresses where the functions can be corrupted and the return may be improper etc so defiantly it is unpredictable and the fourth type visual will be a software exceptions you have seen machine exceptions where the process itself generate the exception for a internal accesses which is predefine.

In this case the software exceptions the program that contain code to check the stack if that is there then we have software exceptions so the basically the purpose of stack over flow is to guide for determining whether the software design should implement production mechanism should be stack over flow basically it tells what is being design whether the stack flow is going to happen or not so this is very important typically the industry.

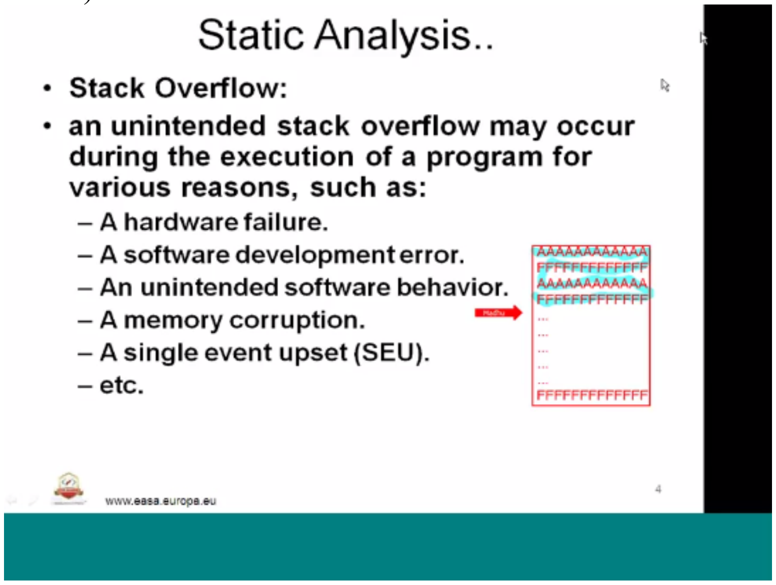
Where the follow stand such as deviant should be so there is a deviant should has be guidance is the stack usage and stack over flow in the areas of code reviews and requirement based verifications however it does not cover the possibility that the data corruptions may occur and lead to stack over flow so the is very important dynamically when we execute to have this consider in terms of data corruptions.

For the purpose stack analysis for the stack over flow is to provide details of specific aspects of that issue then stack over flows so it should be attended appropriately okay and so some times in different in a different program they will have a secondary stack for a additional methods so the secondary, the secondary stack are use less frequently they are used less and their used is under the control of run time functions.

Stack checks are often put in places to monitor the all secondary stack that means the program is already in the embedded software which can be monitored as well with the help of that the data reports we can monitor and arrive it whether the stack is fine there is over flow like this so this is some like a run time check on the primary task, primary stacks so and as well as the secondary stacks so this can be done with the help of review of the, and the map file and over flow should be addressed by requirement based that thing.

As well as the integrations test basically this done during the integrations test it could be software, software integrations on the target actual target this will be performed that stack analysis and stack over flow will be performed at the actual target so analysis of the theoretical worst case scenario helps to determine whether the use and the implementations of the aspect as

be well design in order to adequate manage data that is least to be handled through stack that means that the worst case scenario.
How much it is going to take what is the amount of time and memory it is going to achieve so that also it will help in terms of analysis of the stack so stack analysis of worst case stack rules that is going to be define all the functions and program and obtaining their stack frame size so we need to know all the and their stack size we need to collect it and stack size means stack frame size and also we need to determine the call who is calling and with whom it is functions what is the cal tree analysis so based on the determinations cal graph.
For each separate executions that means the different parts that is nothing but the task so the stack analysis will be done and we need to combine all the data during the worst case stack size executions so this will be definitely done in the souse code level so basically by counting the size of, all size of the perimeters so you do not to need to do manually heir are tools that will report the size so the size need to be analyses manually okay.
(Refer Slide Time: 25:23)



Static Analysis..

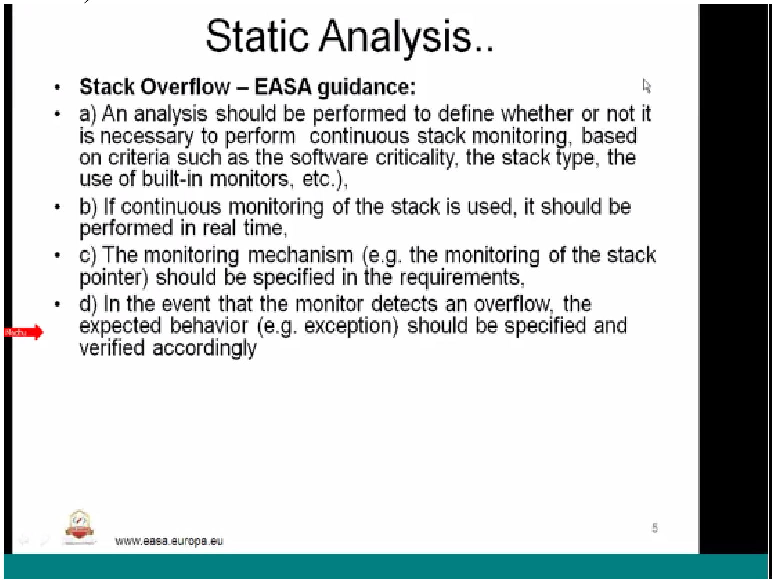
- **Stack Overflow:**
- **an unintended stack overflow may occur during the execution of a program for various reasons, such as:**
  - A hardware failure.
  - A software development error.
  - An unintended software behavior.
  - A memory corruption.
  - A single event upset (SEU).
  - etc.

www.easa.europa.eu

4

The next one is on stack over flow another set of information an unintended stack over flow may occur during the executions of a program for various reasons now we have see the issue that is going to cause of this stack over flow what it can, where as whatever the reason that we are going to have for stack overflow as per the below so they are hardware failure a software development error that means it is not design properly the hardware failure could be memory is memory is corrupted or issue in the parts.
That is not beginning appropriately okay so and the other one could be an unintended software behavior there are software busses which is inherently not return memory corruptions we know that, memory corruptions it could be hardware as well as software so another terms is called single event upset (SEU) so due to some program issues or something there is a one event that is cause everything in terms of squirreling different issue that could lead to a stack overflow that is what it means.
So this is to be done by testing sometime that means will lead to analysis the behavior so typically they take the approach by filling the memory with certain memory pattern and execute

the test which force the maximum message of the stack what will happen is so here is the stack and it is filled with the non patterns such as AAAAAA and FFFFFF it filled in the next line likewise it is filled.
So we know that the stack is filled here then we do some testing make sure that the testing makes the program right the value of this stack into this area that means it is going to be over written by the stack data suppose this much is written or this much is written we know that the program that we have tested as unitized this much that means is still reserved so we read this portions we know that the definite Patten is there and we are able to read it.
We know that portion is not overwritten by the stack data so we know that there is still space available in the stack, and stack has not over flown or we know that how much tack is used so based on that analysis can be done those simple program is going to be easier where as there is a complex programs that for a stack analysis especially in level A software level A are cortical software that they used in the aerospace embedded software.
So in such cases when a stack overflow occurs during the execution of this level A software the consequences that could as be software may be out of control we can predicate definitely we cannot occur to have the aerospace when it flying having a trouble of all this stack definitely analyses appropriately all thou a SCU that is single event upset is unlike to be repeated is single incident to be happen it is an stack corruptions or an overflow occur due to software error then given the same program state and input conditions.
The software will be repeated such reparation may cause the recovery mechanism to be in effected so it is very difficult to recovery such errors which is spitted out of single event actions and followed by the overflow or the software error.
(Refer Slide Time: 30:27)



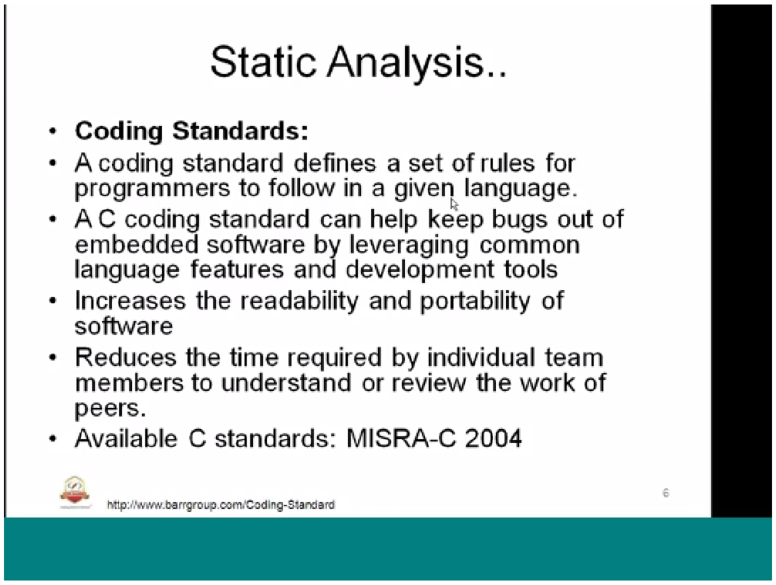
Static Analysis..

- **Stack Overflow – EASA guidance:**
- a) An analysis should be performed to define whether or not it is necessary to perform continuous stack monitoring, based on criteria such as the software criticality, the stack type, the use of built-in monitors, etc.),
- b) If continuous monitoring of the stack is used, it should be performed in real time,
- c) The monitoring mechanism (e.g. the monitoring of the stack pointer) should be specified in the requirements,
- d) In the event that the monitor detects an overflow, the expected behavior (e.g. exception) should be specified and verified accordingly

www.easa.europa.eu 5

So coming to the next slide for the static analysis we have EASA guidance, EASA you know European air space industry so which have is Indian guidelines about quantifying and the quaffing the embedded software for aerospace so their guidelines so as per their guidelines the following four points have to be consider for static analysis.

What are they an analysis should be performed to define whether or not it is necessary to perform continues fact managing based on criteria’s such as software specialty stack type the use of building monitors etc see what will happen is typical embedded software in the air lines industry or aero space programs will have monitoring functions monitoring functions also will have a static monitoring functions and if they are static errors.
If it finds they will be a definite recovery from the static analysis in terms of reaches or the reporting whatever it could be if it is reaching belong certain, likewise so that should be analysis that is what it makes the next one if continues monitoring of the stack overflow it should be performed real time that means if the monitoring part is there in the embedded system within the embedded software the typical embedded software then we should be performed during the real time that is what it means.
The monitoring mechanism example the monitoring of the stack pointer should be specified in the requirements that means the monitoring function is also will be a requirement because without requirement we are not going to have any program that running on the target on field or it will be delivered definitely as per the requirement the monitoring functions will be developed and this monitoring function will monitory the stack continually so this is need to be analysis during run time.
And that is used to be reported is the event of in the event that the monitor dictates and the overflow the expected behavior that means the exceptions or the interpreted should be specified and if there is an issue in the stack such a overflow the same thing that is to be indicated with the program with the expected behavior with raising of the exceptions so that is what we can do are that we know what is the issue for that particular exceptions.
It is because of stack so to conduct monitoring of the stack over flow it is not a difficult one it is basically is a implemented one the additional piece of software it is still worth because the criticality is very important so based on that the requirement will also be laid out and that monitoring policy is also different straggles are followed basically based on the type of stack I mean what sort of stack they are having.
And it is data stack or the program, stack executive level or counters etc so based on that analysis need to be taken care so that is what stack analysis of embedded program will be carried out especially in the aerospace industry that is what the guidelines talks about okay so with that stack over flow we have done.
(Refer Slide Time: 34:51)



Static Analysis..

- **Coding Standards:**
- A coding standard defines a set of rules for programmers to follow in a given language.
- A C coding standard can help keep bugs out of embedded software by leveraging common language features and development tools
- Increases the readability and portability of software
- Reduces the time required by individual team members to understand or review the work of peers.
- Available C standards: MISRA-C 2004

http://www.barrgroup.com/Coding-Standard

6

So next we will move on to the coding standards analysis we are done with the static analysis now we are moving to the next aspect of static analysis that is the coding standards so are very important static analysis that used to be done to you thing that the coding stand is only the coder or the programmer the static analysis.

Has to done on the implementer code or the embedded software has with the help of an independent reviewer whether the standard and the guide lines of the coding have been properly adopted during the programming face or the coding face of the embedded software it is very much important we need to understand so what are the thing that w are going to having coding standards okay.

So coding standard defines the set of rules for programmer to follow, so there are levels of guide lines they give that it could be a guide line it could be rule so in such cases where we have a rules that has to be monitored to be followed I am just trying to write out the guide lines will be there and we have a rules which is very much, guide lines are something like no mandatory but recommend or optional something like that okay.

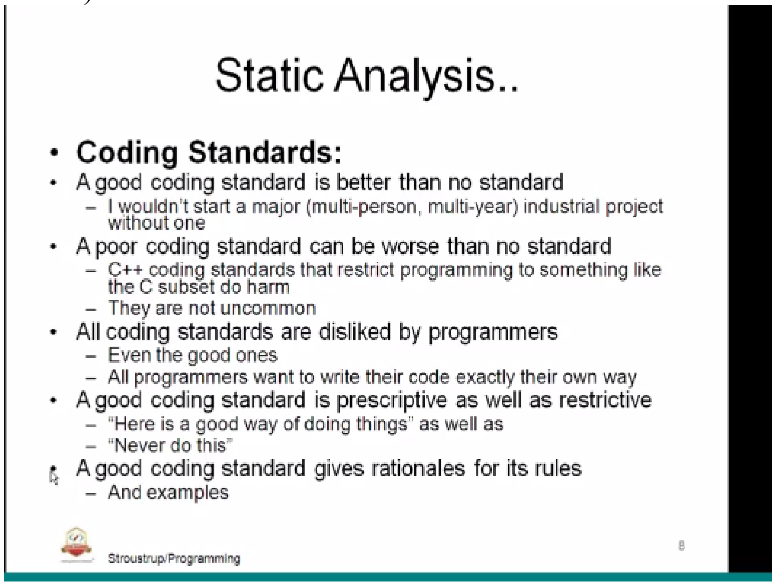
That is to be followed during the programming so programmer could have missed out or it could have ignored or there is a flow while programming due to while last minute work or while doing the rework something would have does not matter if would have warning whatever it is as part of the programming all this need to followed in terms of coding standard and the program may be working on the field it is very fine but definitely.

It has to have a proper way of maintain the code in term of coding standard standards in terms of coding standards that is very important okay the next one is the C coding standard can help keep bugs out of embedded software by leveraging common languages features and development tools definitely there are number of tools that the programmer should try to use whether the programmer is having in terms of coding rules, basically.

Because we may not see the hidden issue of the embedded software which are lied out purely in the coding standard by following the coding standard definitely it will rectify the bugs there are language features which may study, of course samples coding rules we may read and go through

but it is not the scope of the embedded software but all we need to know is what is the coding standard how it will be use and how it will be verified.
That is what we trying to understand and it is important that is also one of the thing okay so basically the coding standards it will be followed properly and it increases the readability and portability of the software that means the software is in a good shape to understand somebody, somebody can read it easily it understanding that means the program is quickly analysis the program can be quickly analysis.
And understandable way it is implemented it is easy to use it or reuse it or you want to a different platform that embedded software or you can scale it or you can maintained by different people or it can be tested easily analysis easily so that is what the basic purpose along with the hidden bugs that could be there when implementing so it reduces the time required by the individuality where as to understand or review the work of peers.
That we know that we are going to have different types of reviews and reviews will be done by independently that means if four people are there in a program who are working as a implementer that four people will be shuffled for doing the, for the implemented program that means person A will give you the person B’s code, person B will review person A’s code so likewise it is easier if and it is going to take less time if the coding standard are followed appropriately because the review’s are quit allot of the programming standard or the coding standard.
So typically embedded industry coding standard they use MISRA-C 2004it ha about hundred plus rules, so the rules are category where the guide lines and the neglecter, neglecter is how have need to be followed and guide lines are something like optional and specifically in the automotive industry they use this MISRA-C okay.
So basically potential benefits of the coding standard people will ignore sometimes they will do a proto typing, whatever it is but basically they need to have the process in plays because the later part of the program is specifically the complex the programmers will be difficult if the standard followed it is not going to cause much if the standard are proper and properly used by the programmer the following.
That is not a expensive definitely they going to save basically because it is going to prevent a bug from kerbing into the code so thus the key strategy for keeping the cost of development somewhere development down is to right code in which the complier linker or the static analysis tool can keep bugs out automatically in other word is allowed to executed better we followed the coding standards of course there are different sources of bug in software programs.
But that is all the next level basically this bugs could have be injected or introduced or created by the original programmer but it could come out as a hidden issue or the bug in the later stage of the program and the program is running for sometimes like a month or a till then it is not reviled so it is difficult to fix, if it found after few months if the programmer was developed and the bug was found after few months typical.
To fix such because the ratability, we do not know where the issue is, typically the number of four and the number of bugs will be introduced by the original programmer can be reduced through visible confirming and certain coding particles especially something like placement or

constant on the left side of each equableness test that means we know equal to and double equal to there is differences.
So programmer have put to as an logic another as an assignment see you would have missed so that lead into a issue and coding standard following that will defiantly use out such issues the other aspects of the coding standard will be commenting we know we use this comments or we use this an adar so it is very important to have a discipline using this of commenting, that means we need to have a consistent comments like the comments should be.
So and so should be very deep and should be within the eighteen lines all this have to be practiced properly so everybody in the organization can easily understand the meaning and the proper use the functions or procedures variables etc so there should be a meaningful name as well as the comments above usually the above they will put it in the C program it depends on what stand you want to follow.
It could be below or the side of the program anywhere but it should be consider and easier okay that is about coding standards in the slide so you will go through some of the coding standards rules for what the code should look like basically here you can see coding standard set of rules what the code should be look like.
(Refer Slide Time: 46:34)



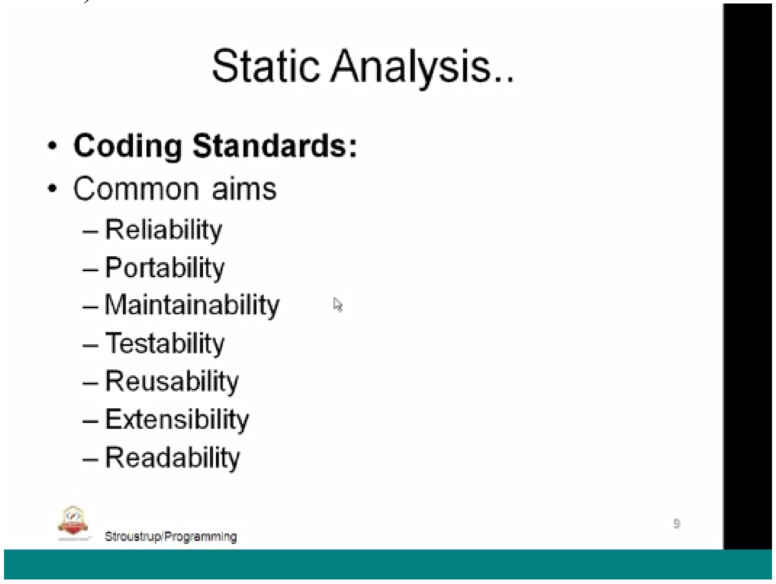
Static Analysis..

- **Coding Standards:**
- A good coding standard is better than no standard
  - I wouldn't start a major (multi-person, multi-year) industrial project without one
- A poor coding standard can be worse than no standard
  - C++ coding standards that restrict programming to something like the C subset do harm
  - They are not uncommon
- All coding standards are disliked by programmers
  - Even the good ones
  - All programmers want to write their code exactly their own way
- A good coding standard is prescriptive as well as restrictive
  - "Here is a good way of doing things" as well as
  - "Never do this"
- A good coding standard gives rationales for its rules
  - And examples

Stroustrup/Programming 8

Indentations rules they call it as for examples use stroustrup layouts it is a famous C++ and C programmer basically. Here return a book and he has put a Indian decision rule basically we can follow that and code we can have typically specifying a sub set of languages that is the better do not use new or through especially C++ provability problem should be there typically specifying the rules for commenting how we should be commenting every functions must have a commenting explaining.
What it does, we know that there should be a proper descriptions and comment for that functions also the lines that are within the functions the programming the programmatic lines often requiring the use of certain regulates that mean, I was saying stdio printer all this we will use it so to avoid the safety issues we have to use it appropriately so we need to follow the rules organization is often try to manage, that means the complicity that we are going to a dilute it

doing the appropriately coding standard so that any of the complex can be easily controlled often they failed and create more complicity then the manage.
So this is what the sources of programming rules talk about continuing on the coding standards a good coding standard is better than no standard we know that, so this also again a statement form this is the one of the binary in C++ and coding standards it could be a multi million project or whatever it is, so without coding standard definitely they are not going to started he says that way but even for the smaller projects.
He say that way but even for the smaller projects any one month two month project are it could be hundreds of planning we need to have standard and process satisfied a poor coding standard can be worse than no standards so that is also another, C++ coding standard that is the programming to some like C sub sets there are not uncommon that is all typical issues so C++ coding standard could restricted C usage because both are incline together all coding standards are disliked by programmers so the good standard also people will the , the programmer sometimes.
But independent ivnet make sure that they all in place, all the programmers want to write their code exactly a good coding standard is prescriptive as well a restrictive that means they should be very prescriptive here is a good way of doing thing as well as never do this that means we should tell both positives as well as negatives in terms of standards, standards which specify both of them registrations or limitations or registrations of what should be done or should not do and prescription also.
We should be tell him what is the good thing to do for that a good coding standard gives rational types, they should be a rational that are the reasons for every coding standard and rules with an examples so that will make easier to maintain and review so that is what the sources of programming recommendations.
(Refer Slide Time: 50:22)



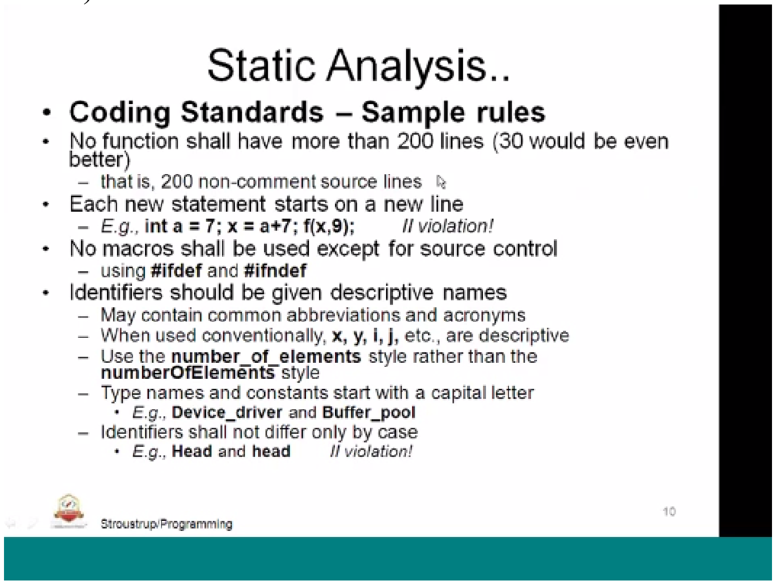
Static Analysis..

- **Coding Standards:**
- Common aims
  - Reliability
  - Portability
  - Maintainability
  - Testability
  - Reusability
  - Extensibility
  - Readability

Stroustrup/Programming 9

And the aims or the goals of the common coding standards as per the below the reliability the portability, maintainability, testability, reusability, extensibility, readability so these are the important aims or the goals of the coding standards I will repeat reliability the portability,

maintainability, testability, reusability, extensibility, readability so this are very important aims for developing the coding standard and following the coding standards.
(Refer Slide Time: 51:10)



Static Analysis..

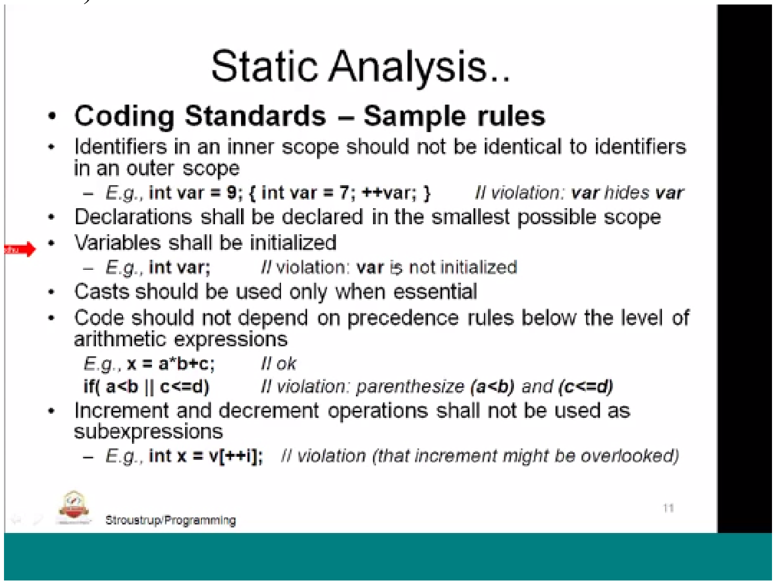
- **Coding Standards – Sample rules**
- No function shall have more than 200 lines (30 would be even better)
  - that is, 200 non-comment source lines
- Each new statement starts on a new line
  - *E.g.*, **int a = 7; x = a+7; f(x,9);** *// violation!*
- No macros shall be used except for source control
  - using **#ifdef** and **#ifndef**
- Identifiers should be given descriptive names
  - May contain common abbreviations and acronyms
  - When used conventionally, **x, y, i, j**, etc., are descriptive
  - Use the **number_of_elements** style rather than the **numberOfElements** style
  - Type names and constants start with a capital letter
    - *E.g.*, **Device_driver** and **Buffer_pool**
  - Identifiers shall not differ only by case
    - *E.g.*, **Head** and **head** *// violation!*

Stroustrup/Programming

10

Few samples rules we will try to go through or coding standards which are observably to for any typical embedded software implementation okay so what are the samples rules no function will have the two hundreds you know why, why because it is very difficult to maintain such program if there is bug while doing the fixing the chances are that could be bug in another places so it is a creation of a new bug while fixing the existing so better to have a program having not more than two hundred lines.
So ideals there is a thirty, fifty lines not more than hundred but definitely not two hundred or more here two hundred non comment sources not that a complete program you can have a descriptions in two three lines so for forty fifty lines you may end of the heading two hundred totally lines but what he talk is about the USE I think I told you about the USE in one of the sessions executive lines of code executive object code.
I will again repeat the statement is there for a semi column is one executive of statement even semi column is also there executive similarly for we have for this barked and in the for barked cons ties one executive similarly while so and what are the contents of if else if is also a it will act as a another executive statement so likewise we cannot have two hundred executive lines in the any functions or two hundred or more each new statement starts.
On a new line we cannot have something like semi column followed by another statement like this we cannot multiple statements in the line that is the one of the strict rule that has to be followed no micro shall be used expect for sources control that means we can afforded to have macros in the program unless we have a sources control that means we if def if and if sort of a thing.
Identifiers should be given descriptive names that means the name should be very much sounding like what it is going to do what is the intermission of that particular variables identifiers use the local variables local variables or any of that so we are going to identify, identifier so that should be small narrative or descriptive in terms of particular signal it could be something like

multispeed so this the variable name so we cannot effort to have a MS people will right M or S so it is not a good particles because even for the programmer himself may be difficult to maintain and execute hence it is better to have a any full name for the for the identifiers so it can contain abbreviations.
'And acronyms when used conventionally x, y, i, j etc are descriptive but better to use number of elements style rather than in number of elements like this so avoid all this single letter things instead of that if you want to mention the number of elements better to use number of elements so that is what you can either way but it should be meaning full and understandable so what he is trying to say is O should be capital and N should be capital should be capital such that it is highlighted properly and read everything to type names.
And constant all with the capital letter that means the name of the constant or a type line always it shout start with the capital where you can see device ever starts with the capital D buffer pool start with the P similarly identifier shall not require only by case so you cannot have head and head and caps and small it is very difficult head is followed by something small head it can be lead or it can lag or whatever it is so better to avoid.
This kind of a violations so all this have to be, why I have put here is the coding standard is have to be in places and there should be analysis in part of the static analysis by the independent tester that what it is and while doing this coding standard static analysis the common objectives such as reliability, portability, maintainability, testability, reusability, extensibility, readability this are achieve that is what the coding standard some more rules.
(Refer Slide Time: 57:13)



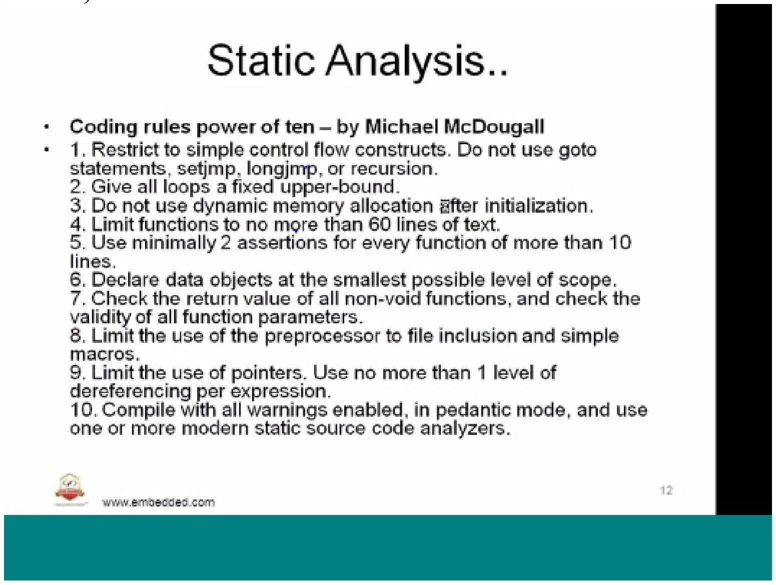
Static Analysis..

- **Coding Standards – Sample rules**
- Identifiers in an inner scope should not be identical to identifiers in an outer scope
  - *E.g.,* **int var = 9; { int var = 7; ++var; }** *// violation:* ***var*** *hides* ***var***
- Declarations shall be declared in the smallest possible scope
- Variables shall be initialized
  - *E.g.,* **int var;** // violation: **var** is not initialized
- Casts should be used only when essential
- Code should not depend on precedence rules below the level of arithmetic expressions
  E.g., **x = a*b+c;** *// ok*
  if(a<b || c<=d) *// violation: parenthesize* ***(a<b)*** *and* ***(c<=d)***
- Increment and decrement operations shall not be used as subexpressions
  - *E.g.,* **int x = v[++i];** *// violation (that increment might be overlooked)*

Stroustrup/Programming 11

We go through identifiers in an inner scope should not be identical to identifiers then of the scope for example we cannot define inner scope variables it is same name it is here, here you can see where it is define outside and inside also same name so you do not know what is going to be used we know what is going to be used but the user or the maintainability perspective it is not easier to maintaining so it is better to have a specific name.
Declaration shall be declared in the smallest possible scope variables shall be initialized should be any variables we cannot just to clear like this we should be declared should be used something

like int where is equal to 0 or depending on initialized value is 0 it should be used next one the cast should be used only one essential so unless otherwise it is needed we should not be use in the casts the type casts code should not depend on parenthesize rules below the level of arithmetic expression.
So we should have dependences so we should be always try to see the implied behavior of the program based on that we need to have parenthesize rules define here for example x=a*b+c, where the multiplications is the parenthesize followed by plus then b+c will happen first and other will multiple that is on the way so this is okay so here in those case you see it is and violations.
You can see parenthesize if a less then b or less than or equal to d defiantly it will going to be violation so other than that you batter so that will avoid the ambiguity as well as some of the compliers what will happen is the presidents could not have been default if and it may take a different way.
So to it is definitely better to have a president taken care in the program so that is be analysis appropriately typically issues that I have seen embedded industry president rules have to be followed appropriately especially this is useful in arithmetic expressions as well as logical expression increment and decrement operations shall not be used as a sub expressions that means we cannot have int x =v and we h=can be array as a index that is being use as a sub expression so that increment might be over look so we do not know what is the end table behavior of this avoid this and outside you do them, so that is what the coding standard samples is about okay.
(Refer Slide Time: 01:01)

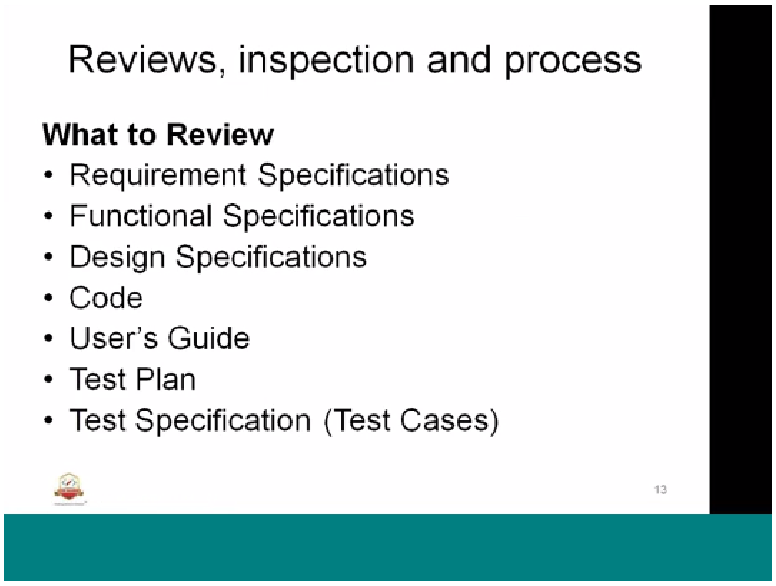


Static Analysis..

- **Coding rules power of ten – by Michael McDougall**
- 1. Restrict to simple control flow constructs. Do not use goto statements, setjmp, longjmp, or recursion.
  2. Give all loops a fixed upper-bound.
  3. Do not use dynamic memory allocation after initialization.
  4. Limit functions to no more than 60 lines of text.
  5. Use minimally 2 assertions for every function of more than 10 lines.
  6. Declare data objects at the smallest possible level of scope.
  7. Check the return value of all non-void functions, and check the validity of all function parameters.
  8. Limit the use of the preprocessor to file inclusion and simple macros.
  9. Limit the use of pointers. Use no more than 1 level of dereferencing per expression.
  10. Compile with all warnings enabled, in pedantic mode, and use one or more modern static source code analyzers.

www.embedded.com 12

So we will continue the coding standard rules, coding rules power and the expression and also.
(Refer Slide Time: 01:24)



Reviews, inspection and process

What to Review

- Requirement Specifications
- Functional Specifications
- Design Specifications
- Code
- User's Guide
- Test Plan
- Test Specification (Test Cases)

13

we will study about the reviews inspections and process in the next class so today we study about the static analysis, over flow coding standards and objectives of the coding standards so that is the end of the session for today.