MISRA C:2012 Workshop

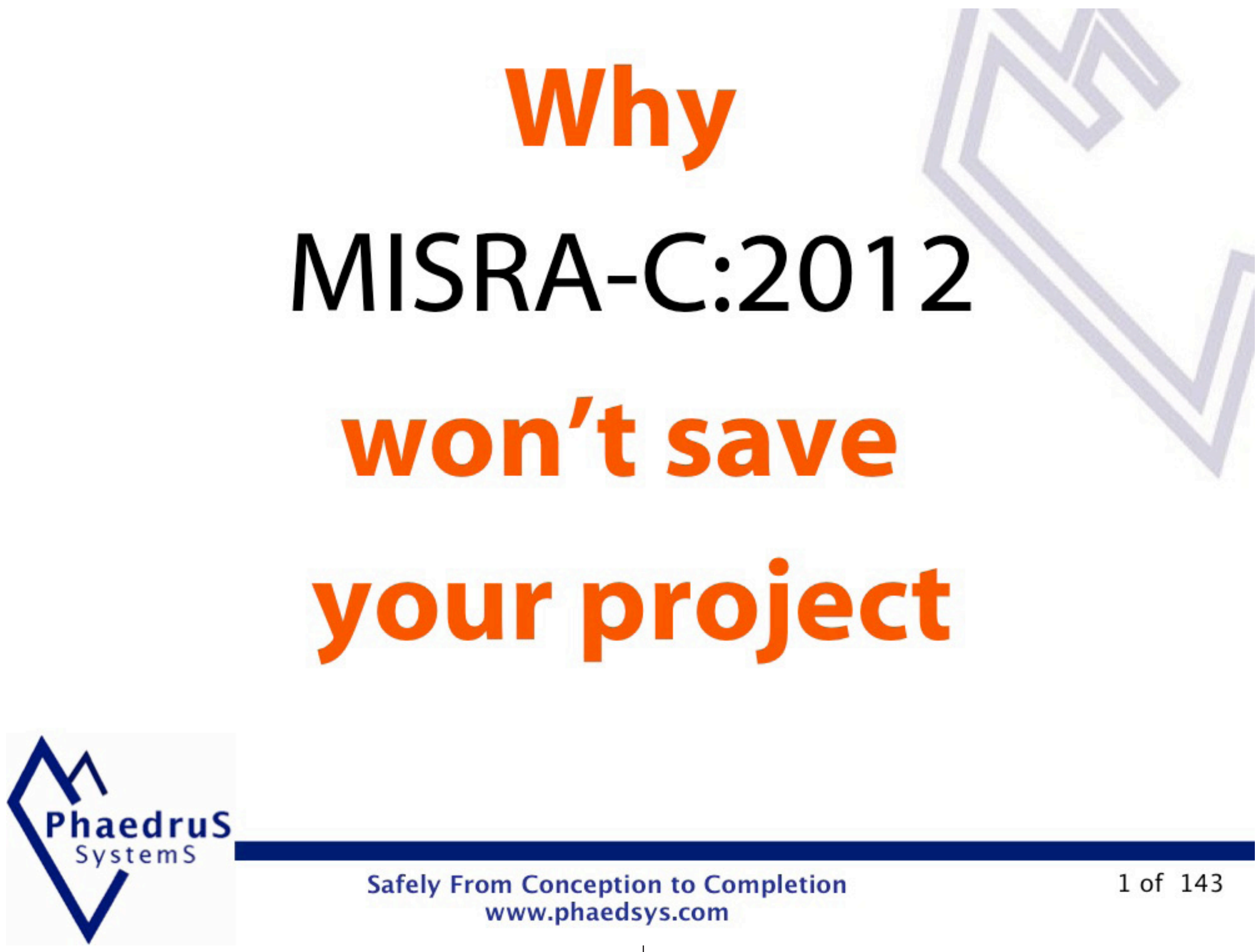
Device-Developer Conference

May 2013

First Edition
by
Eur Ing Chris Hills BSc (Hons),
C. Eng., MIET, MBCS, FRGS, FRSA

Phaedrus Systems

The Art in Embedded Systems
comes through Engineering discipline.



Why MISRA-C:2012 won't save your project

Safely From Conception to Completion
www.phaedsys.com

1 of 143

For a lot of you MISRA-C will not make a lot of difference to the project overall. If misused it could actually make things worse.

Unfortunately many spend more time misusing MISRA-C than using it properly. Once you understand what MISRA-C is and, more importantly, where it fits in the process the rest will fall into place. Then MISRA-C might actually save your project.

MISRA-C:2012

Won't save your project…

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

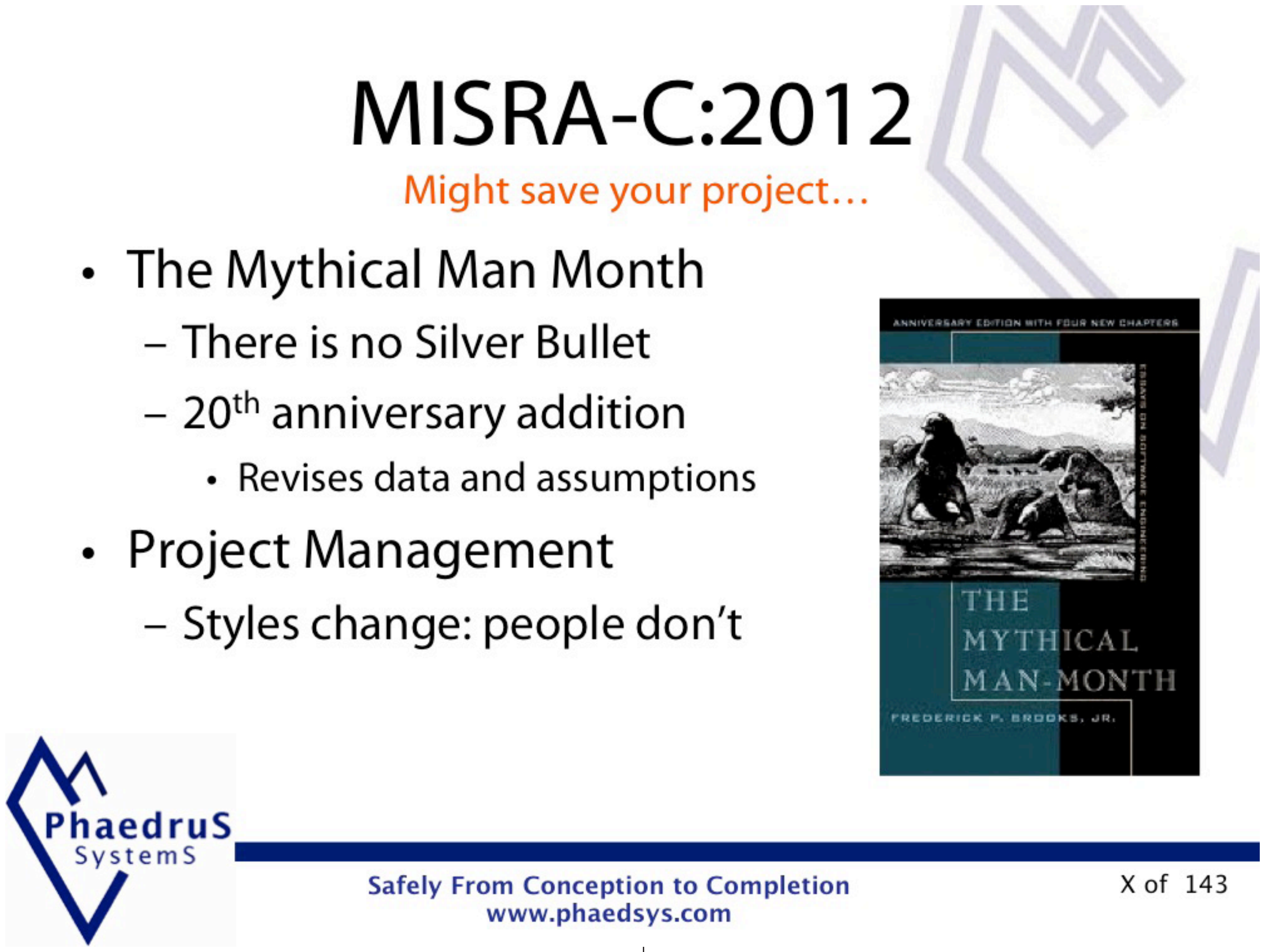
2 of 143

MISRA-C:2012 is NOT a silver bullet. It is not a magic answer.

In fact there are no magic answers unless you live in fairy land or bring your fantasy role-playing games to work. There are far to many who see various tools or methods as The Answer.

As with all tools and methods it is how the tools, methods and processes are used and, more to the point, how they are used in relation to the other tools and the process in general. See Books Mythical Man Month on next page.

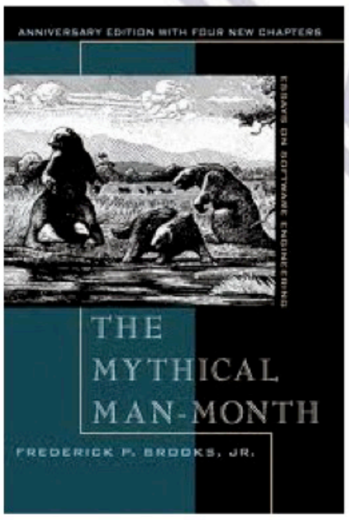
There are no easy answers other than doing it properly.



MISRA-C:2012

Might save your project…

- The Mythical Man Month
  - There is no Silver Bullet
  - 20th anniversary addition
    - Revises data and assumptions
- Project Management
  - Styles change: people don't


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

X of 143

This is a seminal book on project management. The Mythical Man Month says that if it takes 1 man 12 months to do something it does NOT mean that four men can do it in three months. Actually adding people can even extend the time.

Life is more complex than simply dividing peopleinto months, but surprisingly not that much more complex and most of the rules are well understood. This is certainly the case in other disciplines, if you bother to look for them.

The big problem is usually you add more manpower when it is far too late. The damage has been done and the additional people are fire fighting. The answer is to put resources in earlier so you don't get the fire in the first place. The earlier you add the resources Adding more resources earlier, though, will still add up to a lot fewer resources than are usually required later to fire fight.

Mythical Man Month ISBN-13: 978-0201835953

Brooks web site
http://www.cs.unc.edu/~brooks/

http://en.wikipedia.org/wiki/The_Mythical_Man-Month

http://javatroopers.com/Mythical_Man_Month.html

Mythical Man Month Chapter 2:
http://www.cs.virginia.edu/~evans/greatworks/mythical.pdf

MISRA-C:2012

Won't save your project…

Phaedrus Systems

Safely From Conception to Completion
www.phaedsys.com

4 of 143

In many cases requiring MISRA-C conformance is like handing this man a plaster and saying "there there!"

In fact that is what many people do. They don't address the principal problems. More to the point they look at a few details, normally the MISRA-C headline rules rather than the whole process.

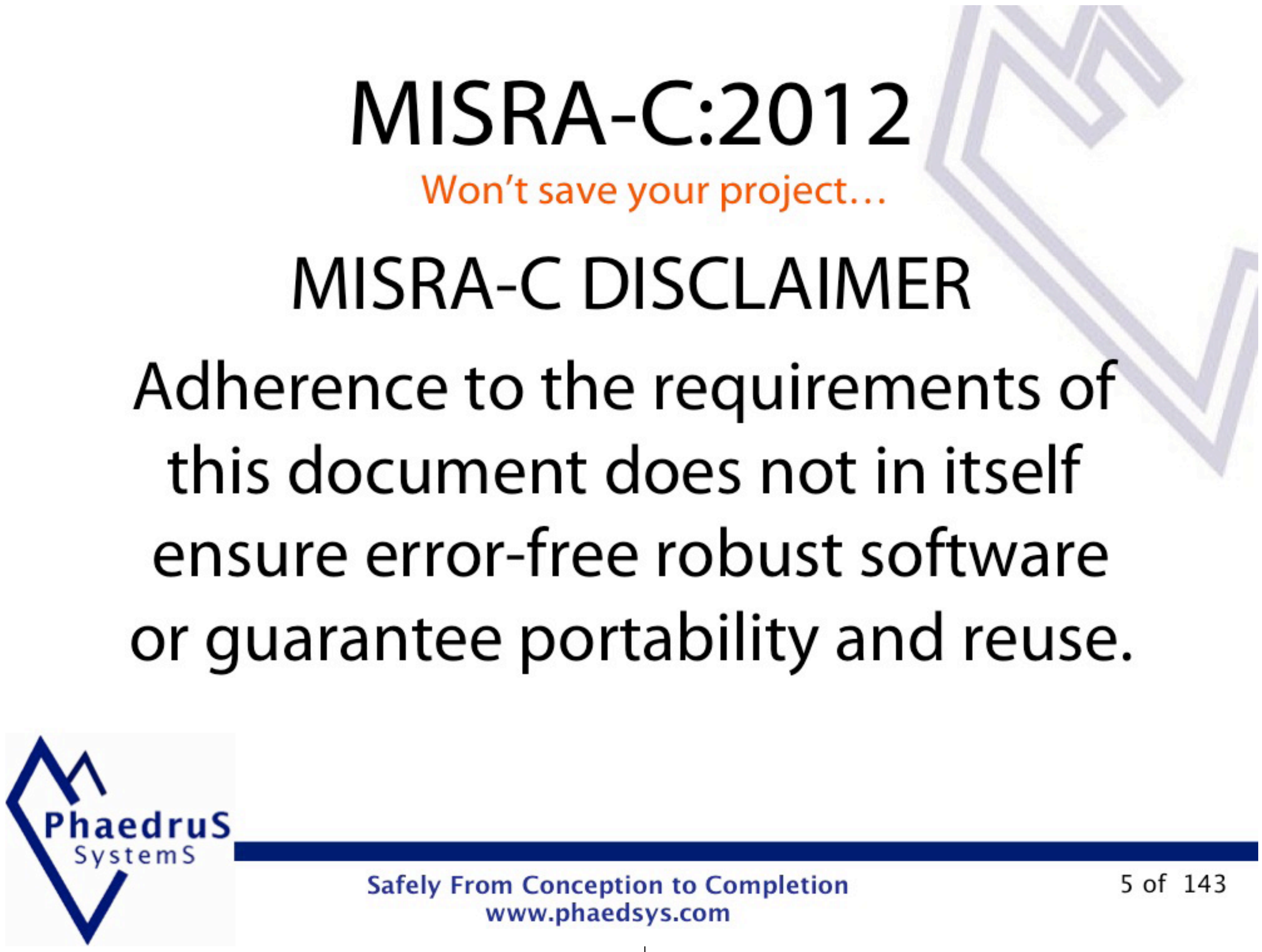
Worse still, they think that by using the Magic Plaster of MISRA-C this man will recover and be dancing with the girl of his dreams in 2 days time aas in a Hollywood film and their software project will equally make a miraculous recovery. Life just isn't like that.

The big problem is the false sense of security people get by using various talisman. In this case applying a plaster and thinking all is well means that this patient will be DEAD ON ARRIVAL.

As we will go on to show using MISRA-C inappropriately and/or on it's own and thinking "all is well" may well mean the project never makes it to the launch. In some serious cases the company may not make it to the next project launch.

One thing you can't do, as one company I know tried to do: buy a MISRA-C checker and run that over the project with the intent of making the code "MISRA Compliant" without reading MISRA-C or configuring the tool.

The problem is, that to make code MISRA Compliant you actually have to have a copy of MISRA-C, read it and then decide which rules you will implement and which deviate, then configure the *static analyser* you are using BEFORE you run a MISRA-C checker over the code.



MISRA-C:2012

Won't save your project…

MISRA-C DISCLAIMER

Adherence to the requirements of this document does not in itself ensure error-free robust software or guarantee portability and reuse.

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

5 of 143

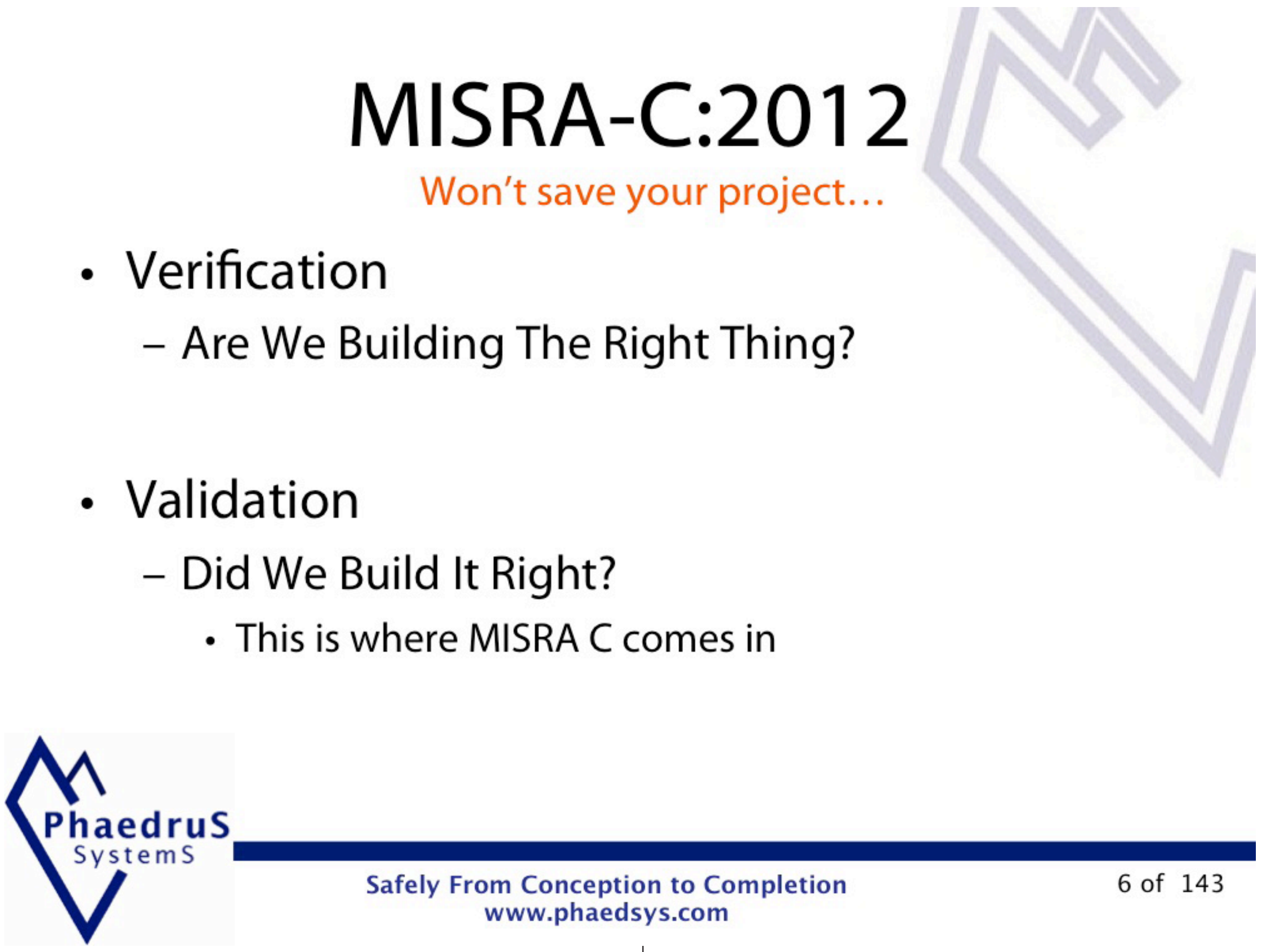
This disclaimer in MISRA-C:2004, and less prominently in MISRA C:2012, should be printed as a poster on the office wall of the development team. This is one of the pages at the end of this document, that may be printed as posters so that you can do this.

Without care, thought, discipline and careful implementation, nothing is automatic and easy. Even the easy and automatic things need to be thought about and understood before being carefully implemented and properly used. You can automate things incorrectly.

Indeed I have seen fast memory checkers where, because the memory write and read was not declared as a "volatile" variable, it had been optimised out and so the memory test did not write or read from memory!

The point is no one thing will guarantee error free, robust, code or indeed a robust or error free system…..

Embedded software rarely exists on it's own. It is part of a system that does something. As with most things you have to look at the overall system which should be greater than the sum of its parts.



MISRA-C:2012

Won't save your project…

- Verification
  - Are We Building The Right Thing?
- Validation
  - Did We Build It Right?
    - This is where MISRA C comes in

Safely From Conception to Completion
www.phaedsys.com

6 of 143

Verification and Validation - that well known double act. Everyone goes on about validation, testing etc. - static analysis, dynamic analysis, unit test etc - they can save a lot of time and effort. In fact they are essential but not on their own.

Static and Dynamic analysis can prove the code and functionality are correct, but NOTE - correct code and correct functionality are two different things.

Static analysis alone can remove many bugs and misuses of C but it cannot prove correct functionality.

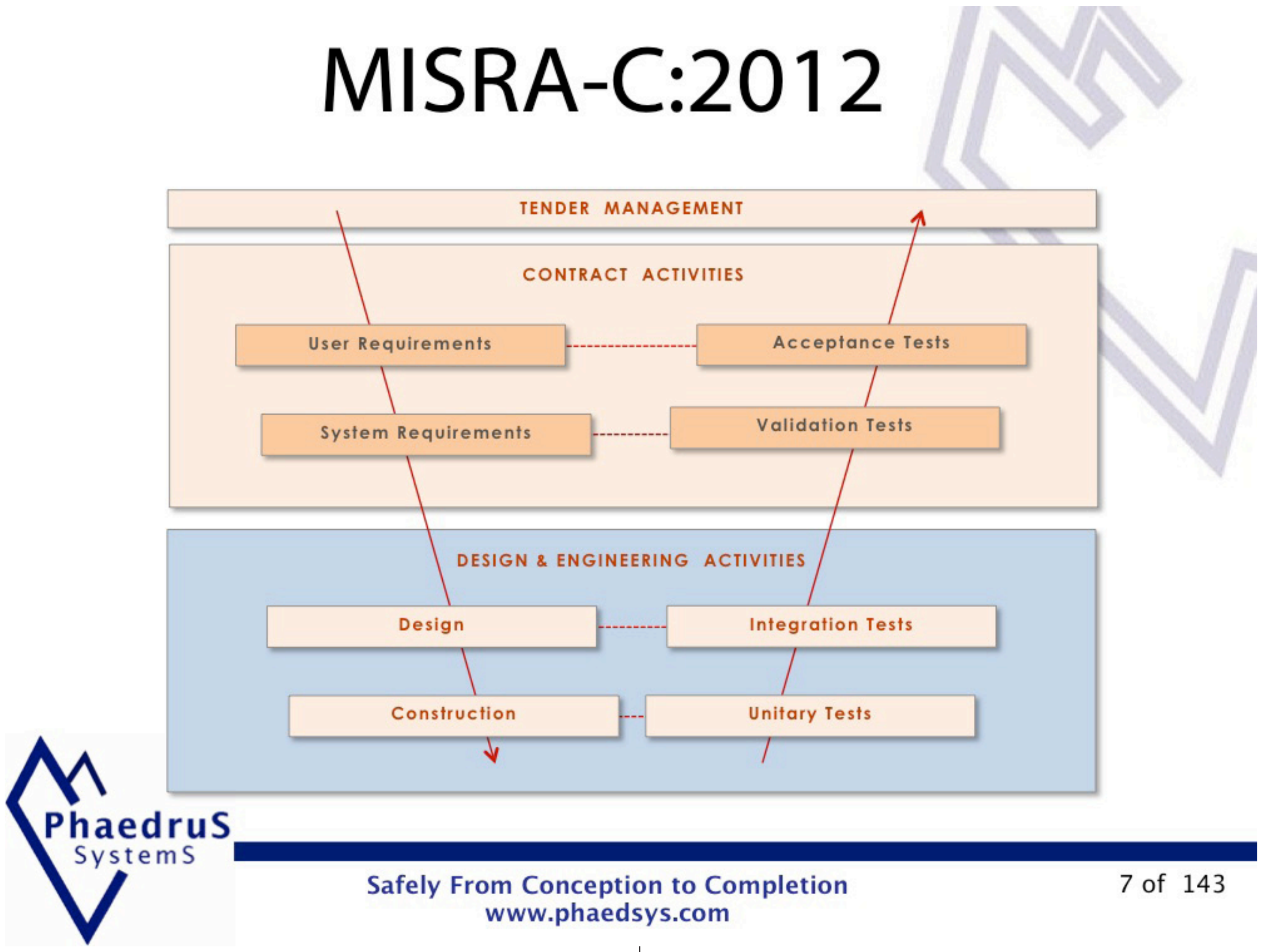
Unit test can prove the low level design correct, but not find many/any bugs in the code and not find if the implementation meets the overall system requirements. So you need both static and dynamic analysis in that order.

No matter how good or validated the test tools are, unless you have a solid requirements specification and a reviewed design that does relate to the requirements, you don't really know what you are validating. The code may be correct in itself and "work" but it may not be doing what the what the end user wants.

Verification: Are the requirements correct?

Validation: Static Is the code correct?

Validation: Dynamic does the unit/system function to the requirements?



MISRA-C:2012

Safely From Conception to Completion
www.phaedsys.com

7 of 143

This is the classic V-model. It works. Or rather it works if used correctly, which is the caveat for all processes! There are many safety critical systems running today saving lives or stopping lives lost, all developed using the V model. There are many more non-critical systems also running, that just quietly get on and work, that were developed using the V model. The crucial point is the V model is conceptual and shows information flow. The User Requirements at the top left (the start) also provide the Acceptance Tests (at the top right, the end); both of which should be completed before a line of code has been written.

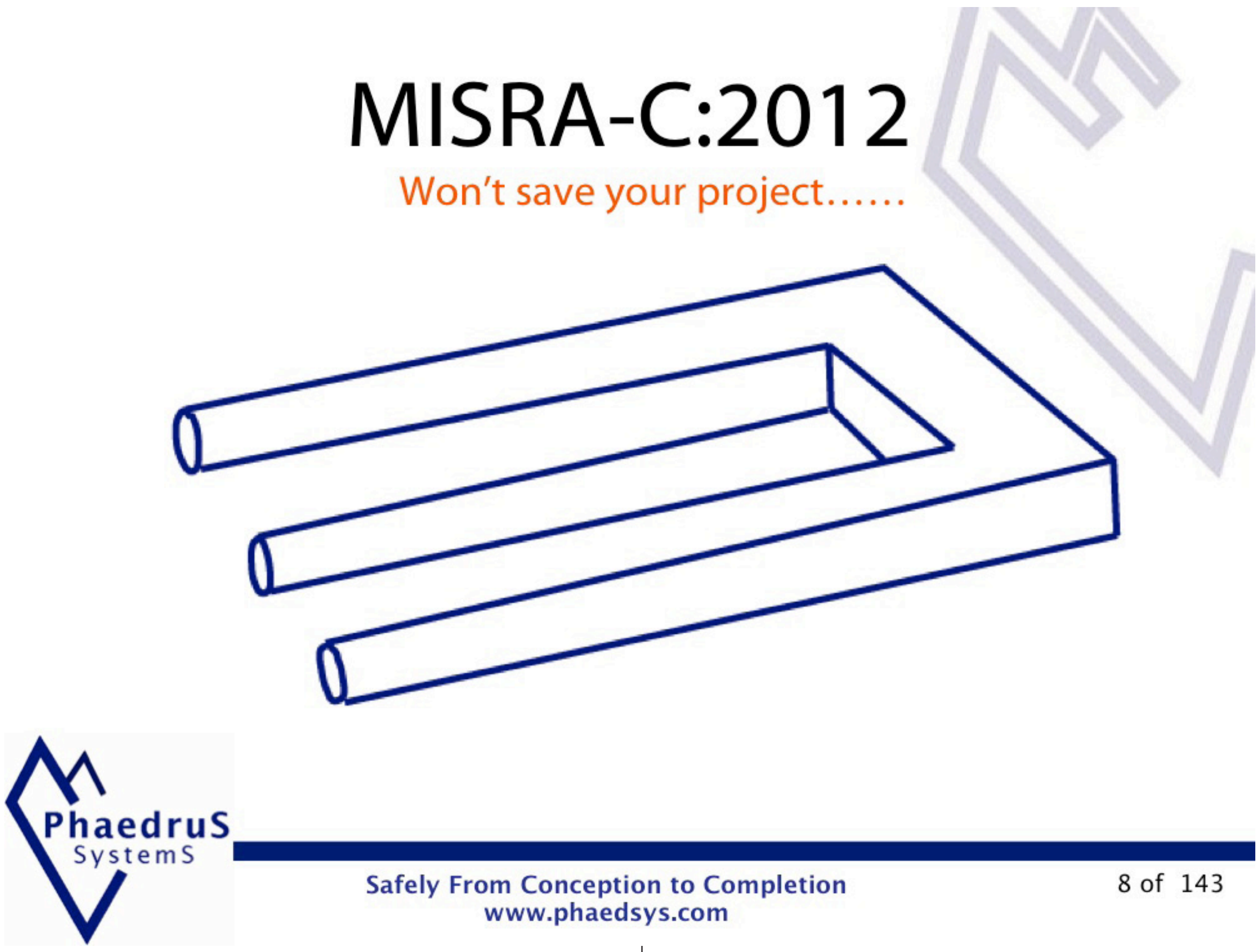
The problem in this model (or any model) are the interfaces - the initial gap between Tender Management and the Requirements - the input to the V that converts a fluffy wish list into hard requirements.

The next interface is the most crucial It is the gap between the pink and blue boxes. The output from the requirements phase is a paper exercise and costs at most a few expense account lunches or buffets when talking to the customers.

When you enter the blue section of design and construction real time, effort and in many cases actual hardware costs start to be non recoverable. Whilst many of you are working in software for embedded systems there is also hardware. The hardware team are actually making physical things that cost money. It is far cheaper to double the time in the requirements phase than start the illusion of progress by writing code and making hardware.

There is the so called "Spin cycle" in the requirements and specification phase. This is where proof of concept and other ideas can be run round. However none of the hardware or software created here should be used in the main development process. (Apart, that is, from 3rd party and other libraries that have already been fully tested and validated.)

Agile is fine for sub contractors on cost plus contracts. You can keep a 9-month contract running for years with continual changes in requirements. It is one of the least cost effective ways of working there is if you are paying the development costs.



MISRA-C:2012

Won't save your project......

PhaedruS
SystemS

Safely From Conception to Completion
www.phaedsys.com

8 of 143

All too often the result is the requirements phase is skimped in the illusion of progress. Then the requirements phase throws out he diagram above. "I can draw it. Why can't you build it?" The common problem here is that:

The requirements phase [TICK}
Has all the documentation [TICK]
All completed [TICK]
Smug look on face of requirements team [TICK]
Engineers told to "get on with it" [TICK]
Failure is fault of Engineering Team [TICK]

Then the Engineers just "get on with it" and several weeks, or months, later problems appear with getting the code to work. Assumptions are made to fill in or paper over cracks. This takes time and effort. Deadlines slip. Pressure mounts. Hacks are used to speed things up. And…. You get the picture.

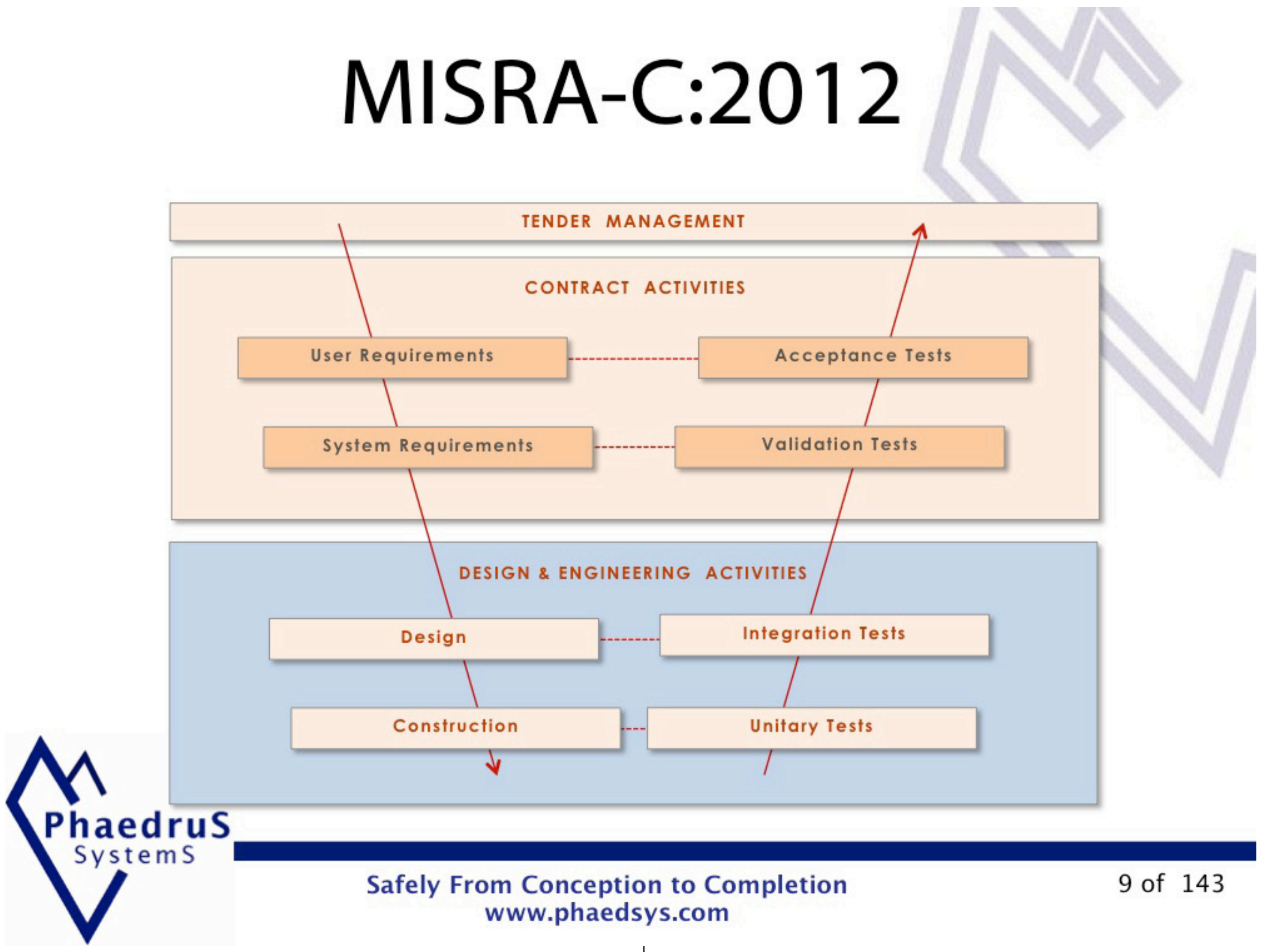
This is even worse when the actual development is out sourced to a different culture, whether in the same country or another.

So what appears to be a problem in the blue implementation phases was in fact generated in the pink requirements and test phase…

Incidentally the drawing would probably pass static analysis and MISRA-C as all the lines are of the correct weight and colour. All the lines are straight or curved as they should be. All lines are complete and are continuous or end at a junction.

This might even pass unit test. Three round ends at one end and a bar at the other…. Dynamic unit tests are micro tests.

Whist software people accept this sort of thing can you imagine what a mechanical engineer would say if presented with incomplete drawings?


MISRA-C:2012

TENDER MANAGEMENT

CONTRACT ACTIVITIES

User Requirements | Acceptance Tests

System Requirements | Validation Tests

DESIGN & ENGINEERING ACTIVITIES

Design | Integration Tests

Construction | Unitary Tests

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

9 of 143

So the output from the pink box, which is the input to the blue box, is the most crucial point. When you try building the item, things don't fit. So you bend them to fit. Then it is YOUR FAULT. What should you do? Question the requirements of course!

Get confirmation ***IN WRITING*** from who ever is responsible. This shift of responsibility back up the process will start to ensure you get accurate requirements.

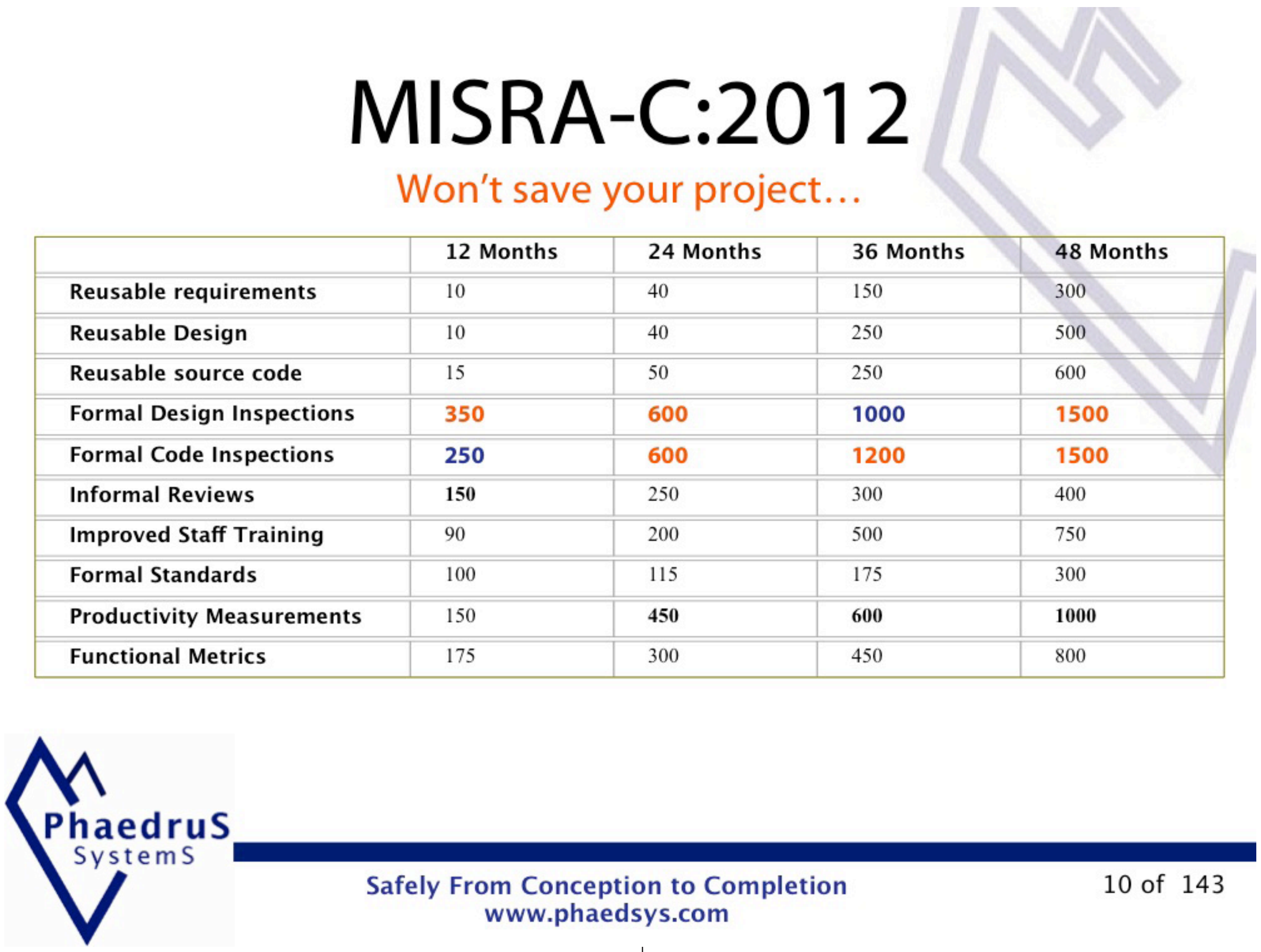
It is ***NOT progress*** if you start developing the wrong thing. Historically successful projects spend longer in the requirements phase than projects that fail! Typically they spend over 50% of the project time in the requirements phase.

Contact Phaedrus Systems for the presentation: Requirements are Required. (email MISRA@phaedsys.com)

The other thing that is often missed is that the ***REQUIREMENTS PHASE GENERATES THE SYSTEM TESTS***

So far from throwing some ideas at the implementation team and running, the requirements team are also specifying the tests so if there are gaps in the requirements there will be gaps in the tests.

You can't test for things that were not required. So where in many projects you get hacks, patches and then delays at the final testing while bug hunting and fixing things, it would have been better to spend a week or two more in the requirements phase. This would have removed the problems before they happened, shortening the overall project time-table.



MISRA-C:2012

Won't save your project…

	12 Months	24 Months	36 Months	48 Months
Reusable requirements	10	40	150	300
Reusable Design	10	40	250	500
Reusable source code	15	50	250	600
Formal Design Inspections	**350**	**600**	**1000**	**1500**
Formal Code Inspections	**250**	**600**	**1200**	**1500**
Informal Reviews	**150**	250	300	400
Improved Staff Training	90	200	500	750
Formal Standards	100	115	175	300
Productivity Measurements	150	**450**	**600**	**1000**
Functional Metrics	175	300	450	800

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

10 of 143

Return On Investment (ROI) per 100 (USD/GBP/Yen) invested in a project. Source: Programming Research.

This chart shows that Formal Design Inspections pay off the most followed by Formal Code Inspections. BUT, don't forget the formal code inspections assume that the design is right!

These two score the highest and second highest ROI in all categories. In fact these two combined have a higher ROI than all the rest put together. Red numbers are best return in each criteria and blue are second best return.

Design inspections pay off faster because if you get the design wrong you are wasting time and effort (money) on building the wrong thing in the next stages.

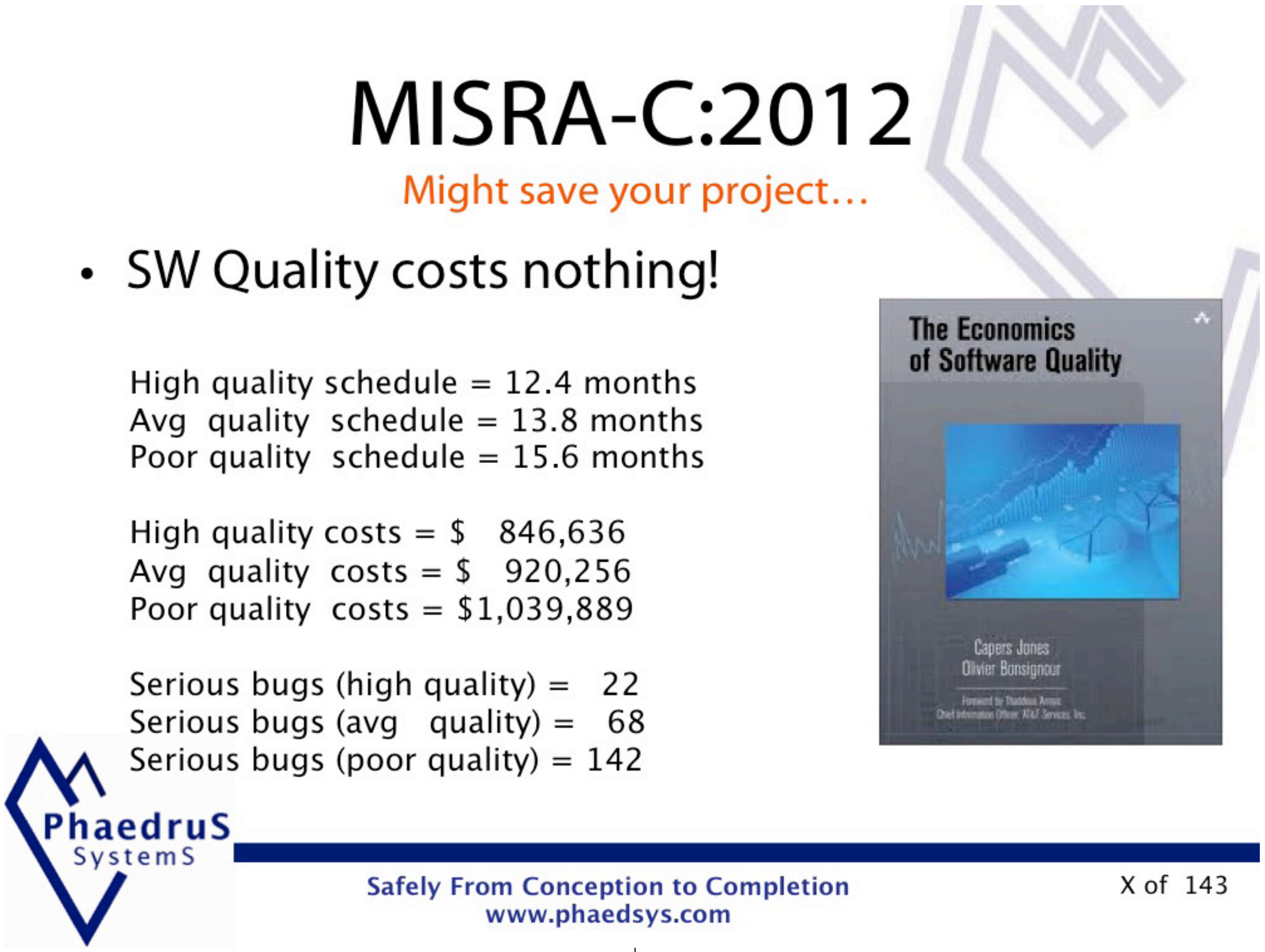
With coding the return is higher the further from coding you get as the costs of fixing a coding bug escalate dramatically the further from the coding phase.

A bug that would cost 1 (USD/GBP/YEN etc.) if fixed in the coding phase (eg though static analysis) could cost 50,000 (USD/GBP/YEN etc.) if it escaped into the field.

I have a real world case where that happened. The company in question had turned down an "expensive" tool solution costing 20K during development. When the tool was demonstrated again, after the problem had appeared in the field and had been fixed, the tool found the "50K bug" in about 15 minutes.

The tool also uncovered another 5 problems, of similar magnitude, that were still in the code that was in the field waiting for the dice to fall and cause a problem.

Look at recent events (spring 2013) with the 787 Dreamliner and you can see the cost of fixing the bug may pale into insignificance compared to other longer term indirect costs.



MISRA-C:2012

Might save your project…

- SW Quality costs nothing!

High quality schedule = 12.4 months
Avg quality schedule = 13.8 months
Poor quality schedule = 15.6 months

High quality costs = $ 846,636
Avg quality costs = $ 920,256
Poor quality costs = $1,039,889

Serious bugs (high quality) = 22
Serious bugs (avg quality) = 68
Serious bugs (poor quality) = 142

The Economics of Software Quality

Capers Jones
Olivier Bonsignour

Safely From Conception to Completion
www.phaedsys.com

X of 143

Here are three samples for schedules and costs: high quality, average, and poor quality. All three are 1000 function points in size. Costs are based on $10,000 per month.

The high quality case used static analysis, inspections, and formal testing.

The average quality case used static analysis and quasi formal testing

The poor quality case used only informal testing. thus cheaper to start.

This book has all the facts and figures to back up these assertions. One of the authors has been involved as an expert witness in many legal cases and had access to the data, true costs etc. (And not just the stories that have appeared in the press.

A useful reference book to have as it is not from a tool vendor and thus is an independent authority. See this Short Video by Capers Jones

http://www.youtube.com/watch?v=zmrqsQxv_yo

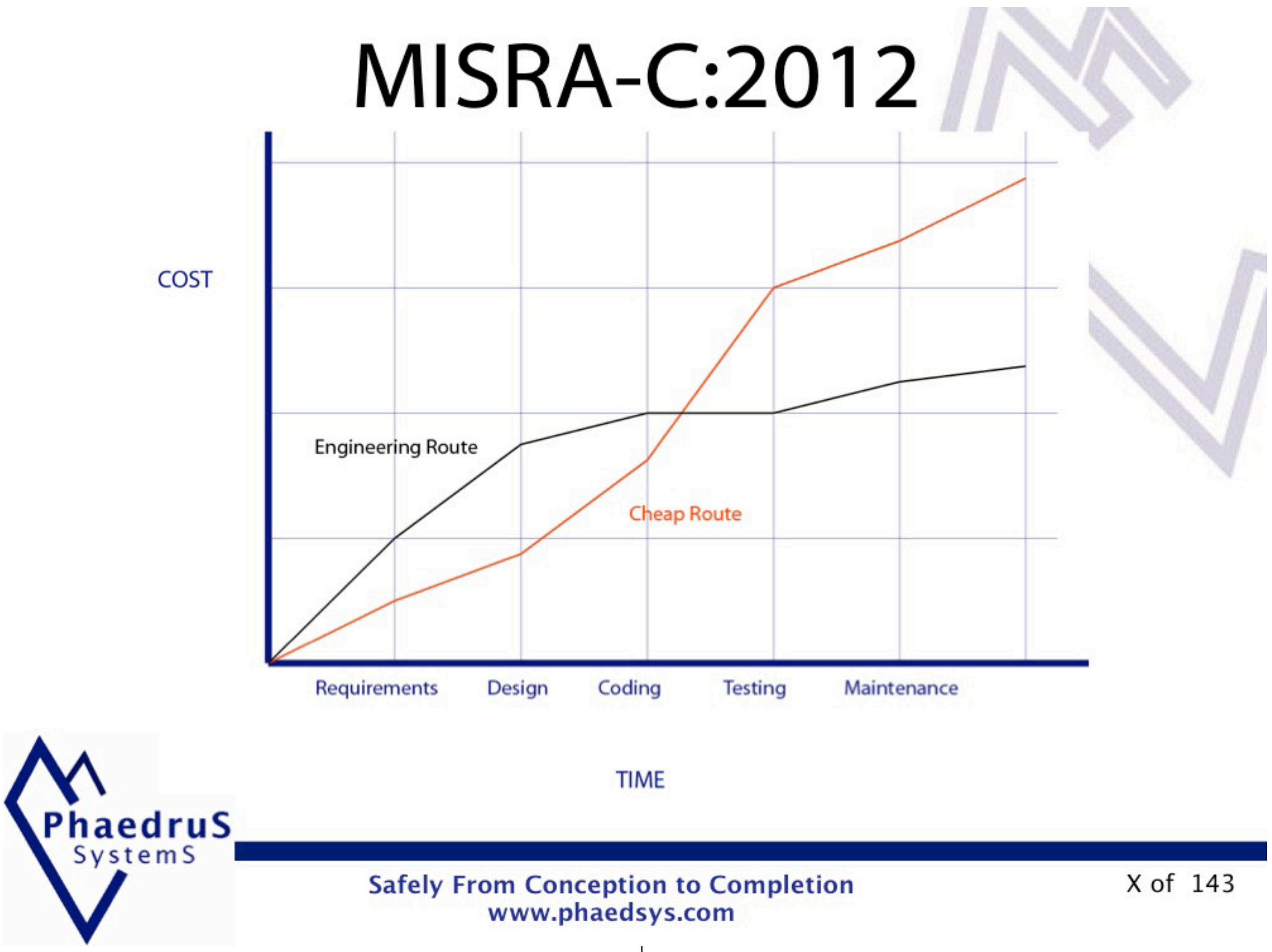
The slide on the following page is at 2:40 into the video

Also worth listening to is a Podcast: Economics of Software Quality - An Interview with Capers Jones The interviewer is Rex Black (also a well known safety systems expert in his own right)

Part 1
http://www.youtube.com/watch?v=zo8JI9MVxQg

Part 2
http://www.youtube.com/watch?v=FLDgRtzq-Cc

http://sqgne.org/presentations/2011-12/Jones-Sep-2011.pdf



Safely From Conception to Completion
www.phaedsys.com

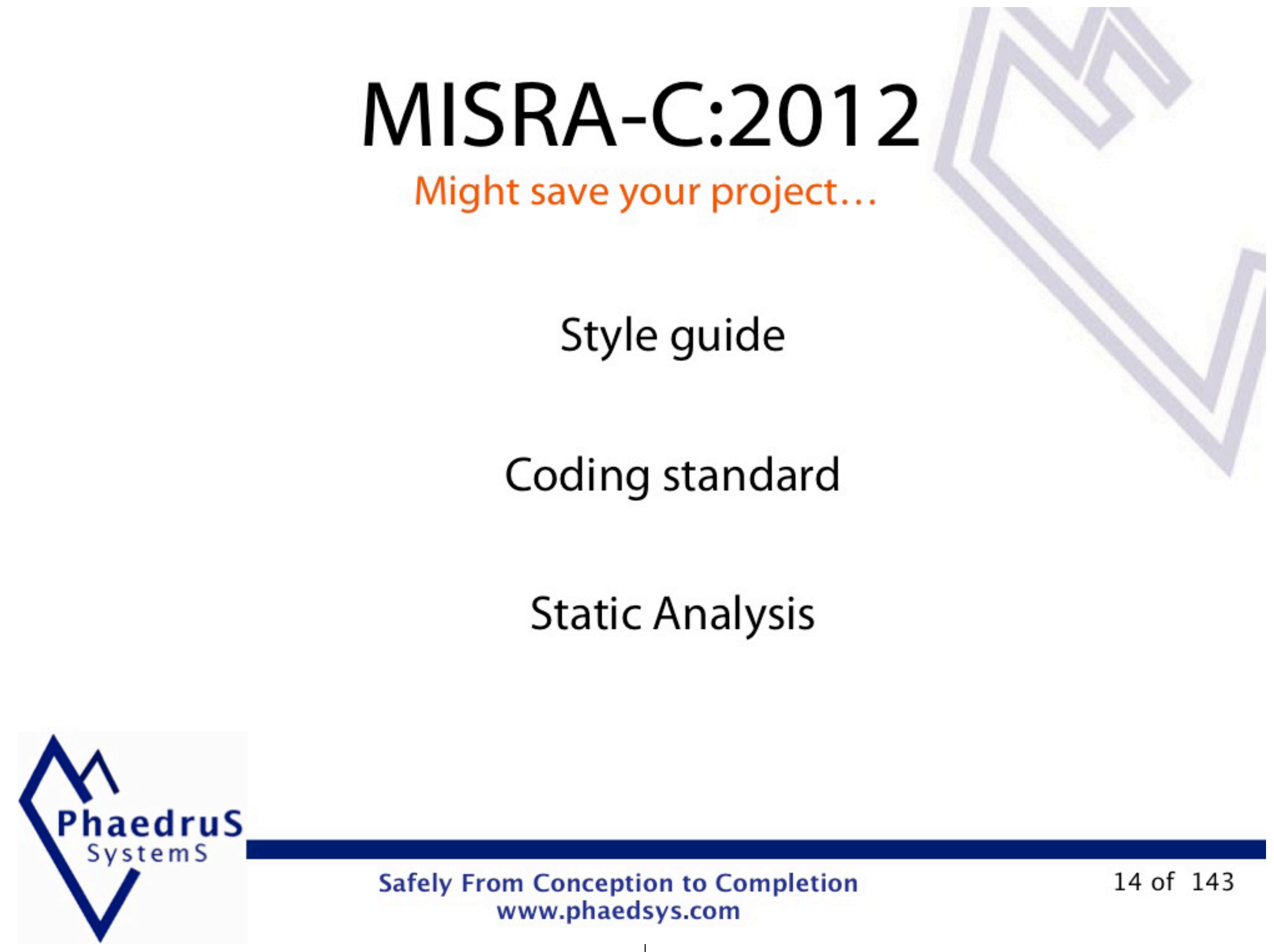
X of 143

This is the slide at 2 min 40 seconds in to the video by Capers Jones at http://www.youtube.com/watch?v=zmrqsQxv_yo

There are similar graphs, from many others, who have done similar work over the last 50 years. As the pool of information and examples grows the numbers used in the reviews are getting larger and the statistics more accurate. In the past, there were hundreds of examples to draw, from now there are tens of thousands and they all confirm the studies.

The "cheap route" costs less to get started. However it then gets ***VERY*** expensive from the coding stage. The problem is that the costs incurred from and after the coding stage start to become exponential. And they have no time limit. Bugs in the field will come back to haunt you. Even if the device is obsolete, or no longer current, its failure still damages the company's reputation even if you don't have to fix the bug.

Engineering route is more expensive to start but costs a lot less in the longer term. Also, by finishing the project on time (with lower costs) you can start receiving an income from sales sooner, and also start the next project.



MISRA-C:2012

Might save your project…

Style guide

Coding standard

Static Analysis

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

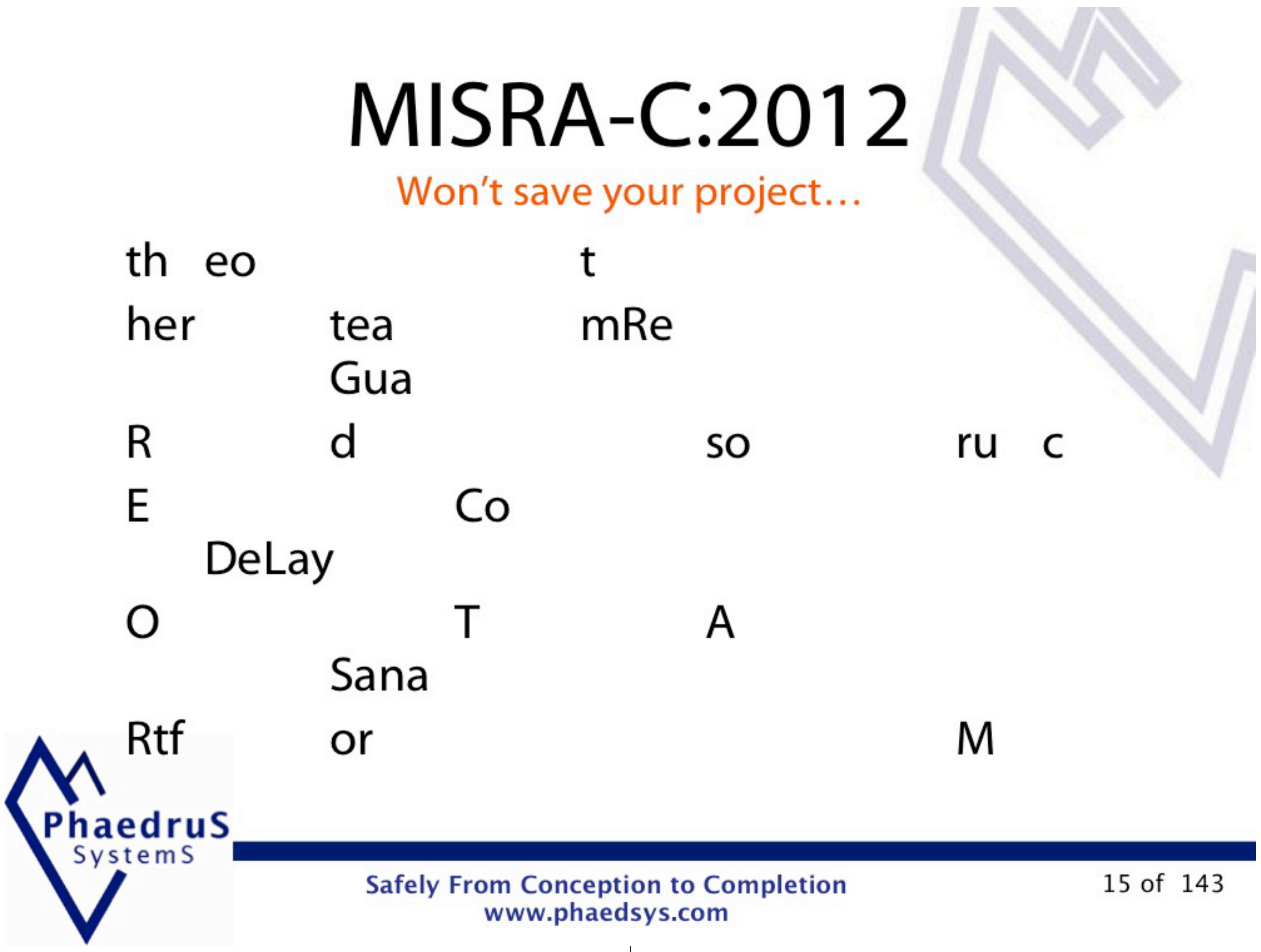
14 of 143

So, after spending time on requirements and design, with formal reviews for both, we get to the coding phase: We will start with Style Guides.

All the code needs to look the same. This helps readability. If the code is uniform things that are wrong ***STAND OUT***.

A uniform style is psychologically good as the brain is not spending effort on working out what is there but can actually look at what the code is doing.

A short demonstration of this follows.



MISRA-C:2012

Won't save your project…

th eo t
her tea mRe
Gua
R d so ru c
E Co
DeLay
O T A
Sana
Rtf or M

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

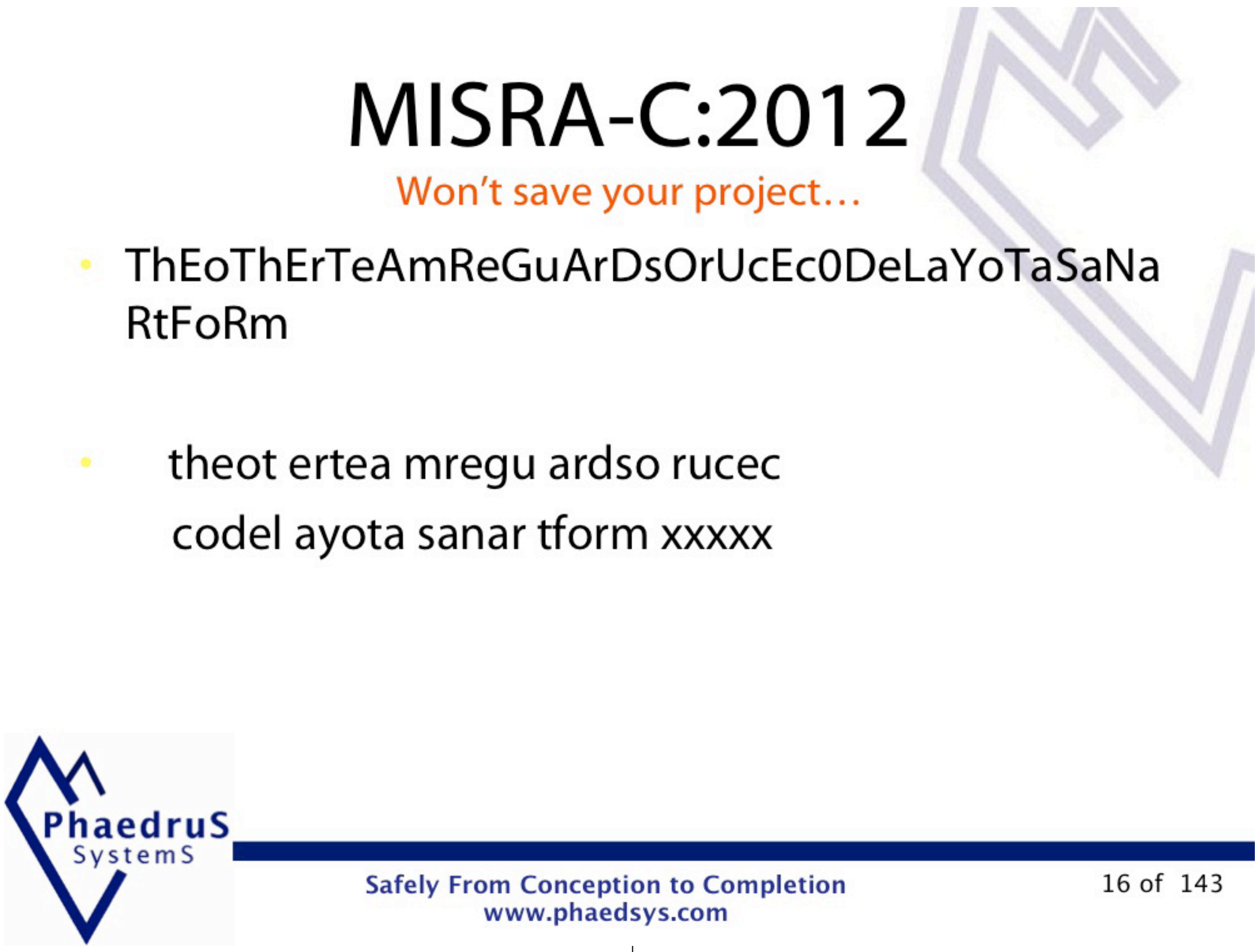
15 of 143

The text on the slide above was sent to me, many years ago by one of my team, exactly as formatted. He had been assisting on a code review for another team.

I normally give 10 seconds for the audience to work it out. 99.99% can't do it in 10 or less seconds which is the point…

NOTE if you can work it out in 10 seconds you are not normal and should not be doing either the style guide or the coding standard!

NEXT SLIDE (before they work it out what it says)



MISRA-C:2012

Won't save your project…

- ThEoThErTeAmReGuArDsOrUcEc0DeLaYoTaSaNa RtFoRm

- theot ertea mregu ardso rucec
  codel ayota sanar tform xxxxx

Phaedrus Systems

Safely From Conception to Completion
www.phaedsys.com

16 of 143

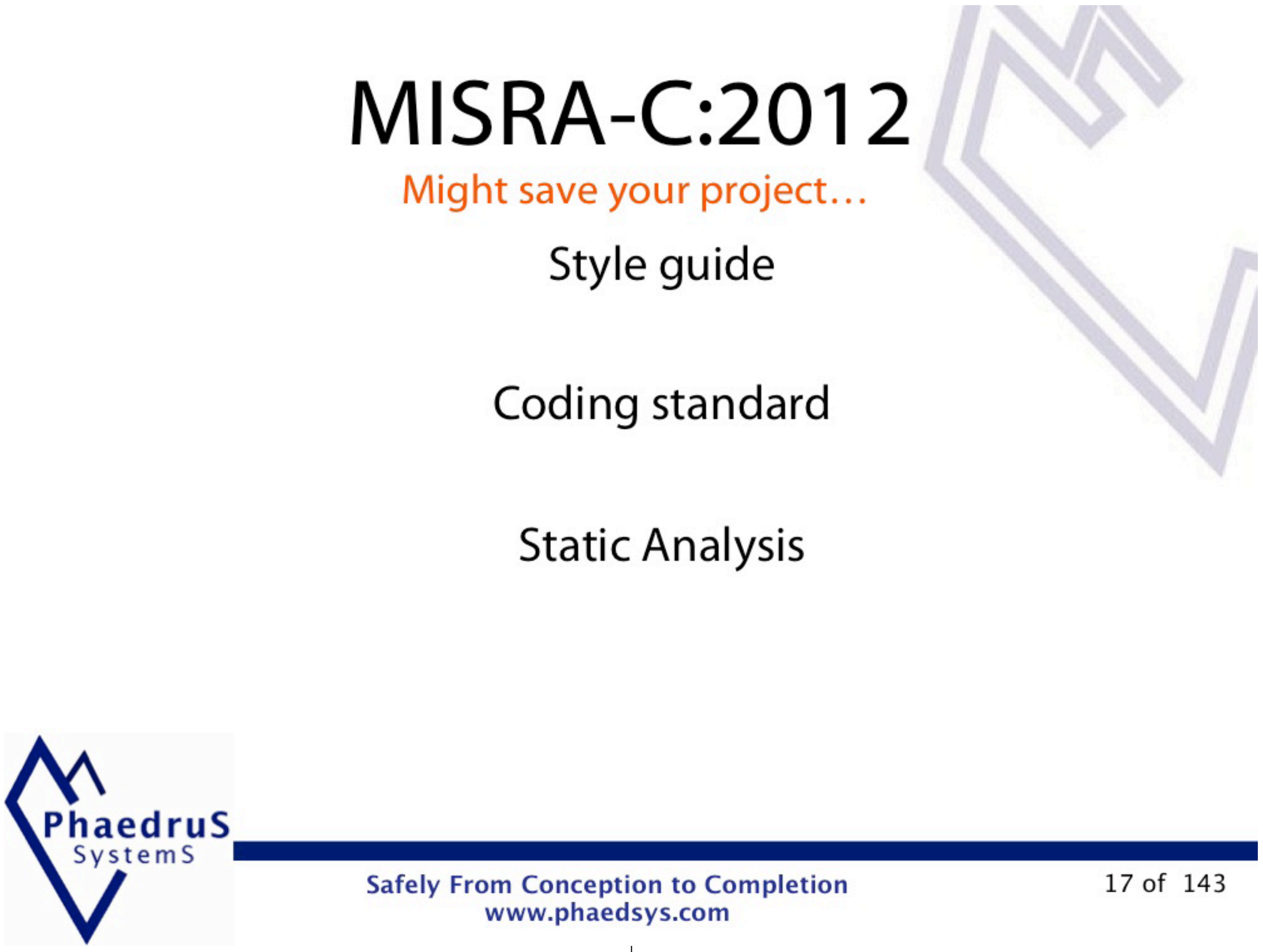
This is the same characters in two different styles! Again I give about 10 seconds for each (before people can work it out! J) What the text actually says is:

The other team regard source code layout as an art form.

The problem was that it took all day to work out what the code was saying due to the multiplicity of styles. It was very inefficient. Also errors are not obvious… I think there is a different error in each of the three examples.

With one style across the whole project and preferably the whole company, less time is wasted on having to sub-consciously "translate" the code as you read it. Software engineers should work to the house style and worry about the more important things - problem solving and algorithms.

Most compiler IDE's will work with templates for code. These cover things such as function header blocks, constructs for switch statements, structures, "for" loops etc, and can be used across the company. There may also be a need for project specific templates. These should be the same as the company wide versions but with the project specific things added. For example, document reference numbers.



MISRA-C:2012

Might save your project…

Style guide

Coding standard

Static Analysis

Safely From Conception to Completion
www.phaedsys.com

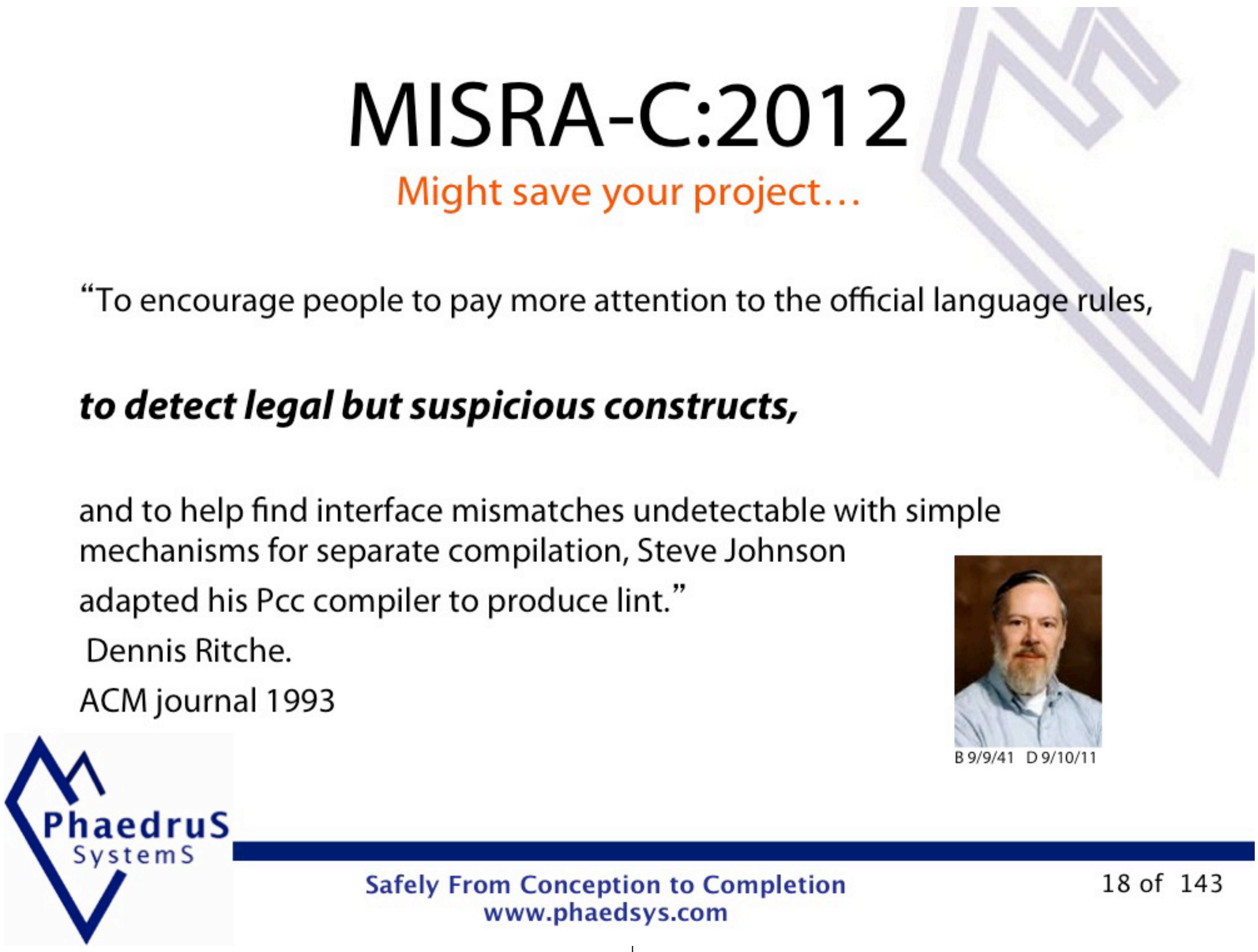
17 of 143

A Style Guide is a MUST! Consistency across the project, and preferably the company, makes sense and removes stress. It also makes code reviews far more effective. Many companies mix the coding standard and the style guide into one document. MISRA-C is a coding standard but, before covering that, I want to look at static analysis.

C is a very flexible language. It is not strongly typed. You can stuff a "long long" into a "char" quite legally, silently losing bytes of data. You can put a negative signed variable in to an unsigned variable, both losing the sign and changing the value if it was negative.

A compiler is a language translator and you will note it is referred to as a "translator" in MISRA-C (and in ISO language standards).

A compiler will pick up syntactical errors but not semantic errors. Even compilers that claim to do semantics are very lightweight when it comes to it. Those that claim to do static analysis are also somewhat suspect, as static analysis requires an engine that is more complex than a compiler.



MISRA-C:2012

Might save your project…

"To encourage people to pay more attention to the official language rules,

to detect legal but suspicious constructs,

and to help find interface mismatches undetectable with simple mechanisms for separate compilation, Steve Johnson adapted his Pcc compiler to produce lint."

Dennis Ritche.

ACM journal 1993

B 9/9/41 D 9/10/11

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

18 of 143

The first static analyser for C (lint) was created, to detect legal but suspicious constructs. According to Dennis Ritchie, "A lot of legal C is dangerous." He wrote that in 1993. He was writing about the first lint program. That was constructed in 1976, before they wrote the seminal K&R book, the first language reference for C, in 1978. Over a decade before ISO C or ANSI-C there were problems with C being misused.

Also programmers like to try and prove how clever they are with C. Brian Kernighan had a comment on that which comes up later in the presentation. It seems that lint (static analysis) was intended to be part of the standard C compiler chain and certainly was in compilers running on UNIX..

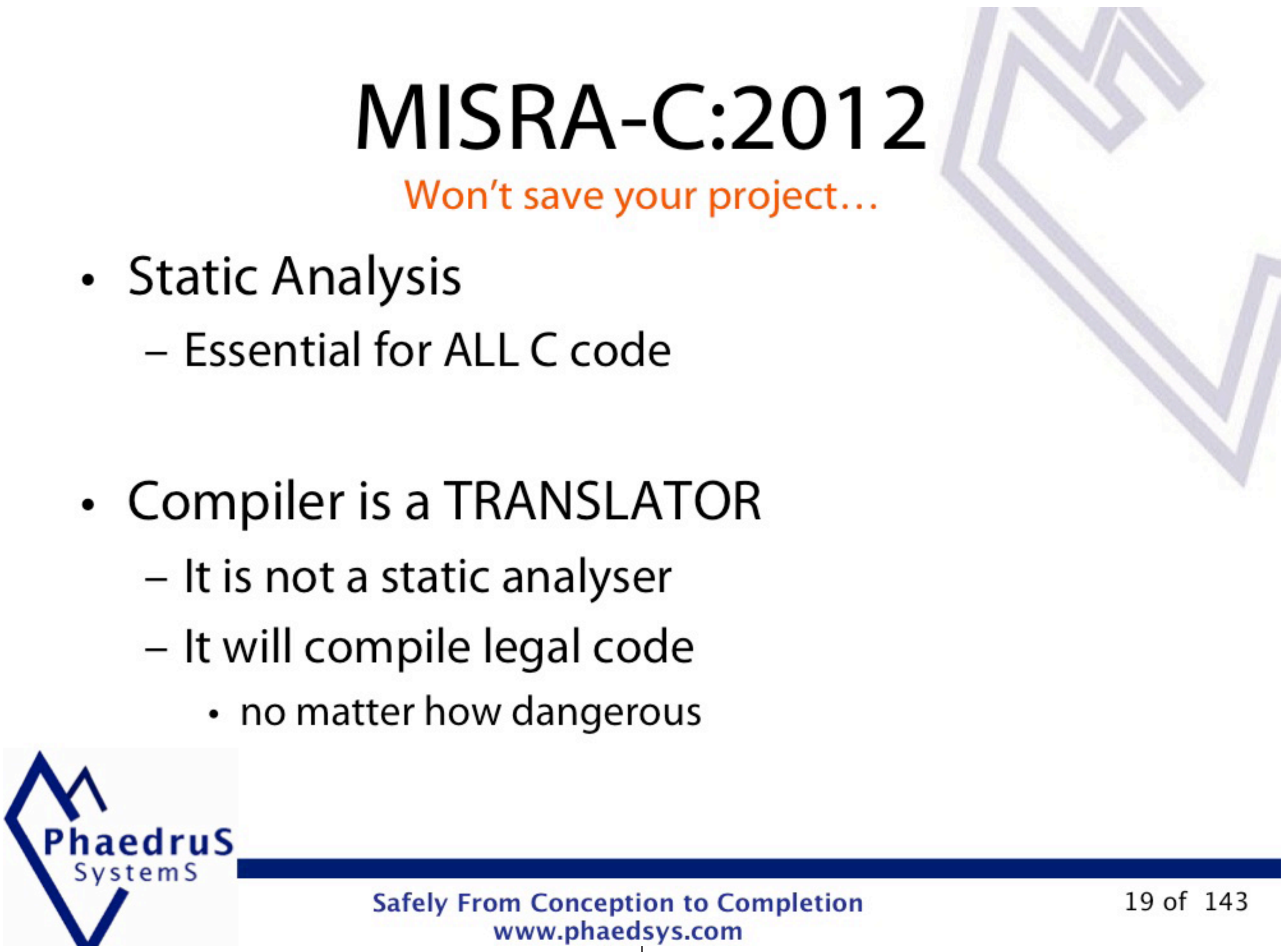
The problem is that it never survived on the leap to the PC development platforms. Many of us did use lint in the 80's but most programmers never started the habit and it seems universities never pushed it either.

The culture of "it compiles so it must be OK!" started to prevail.

Since the original lint, static analysers have developed. At the high end, they are very powerful code analysers that can enforce local coding standards as well as rigorously analyse code with configurations for many dialects of C. In the embedded world most compilers have extensions for the hardware architecture, specific IO and registers.

Check the pedigree of any static analyser you intend to use: static analysers are more complex than compilers - they have far more to do.

The father of static analysis: http://en.wikipedia.org/wiki/Stephen_C._Johnson



MISRA-C:2012

Won't save your project…

- Static Analysis
  - Essential for ALL C code
- Compiler is a TRANSLATOR
  - It is not a static analyser
  - It will compile legal code
    - no matter how dangerous

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

19 of 143

The K&R Team intended static analysis to be part for the compile chain with the original *lint.* Static Analysis saves Time (== MONEY) by finding bugs at the time you write them. C is often called a "write only" language as it is difficult to read later. So fixing bugs whilst you still remember how and why you wrote the code is a good idea..

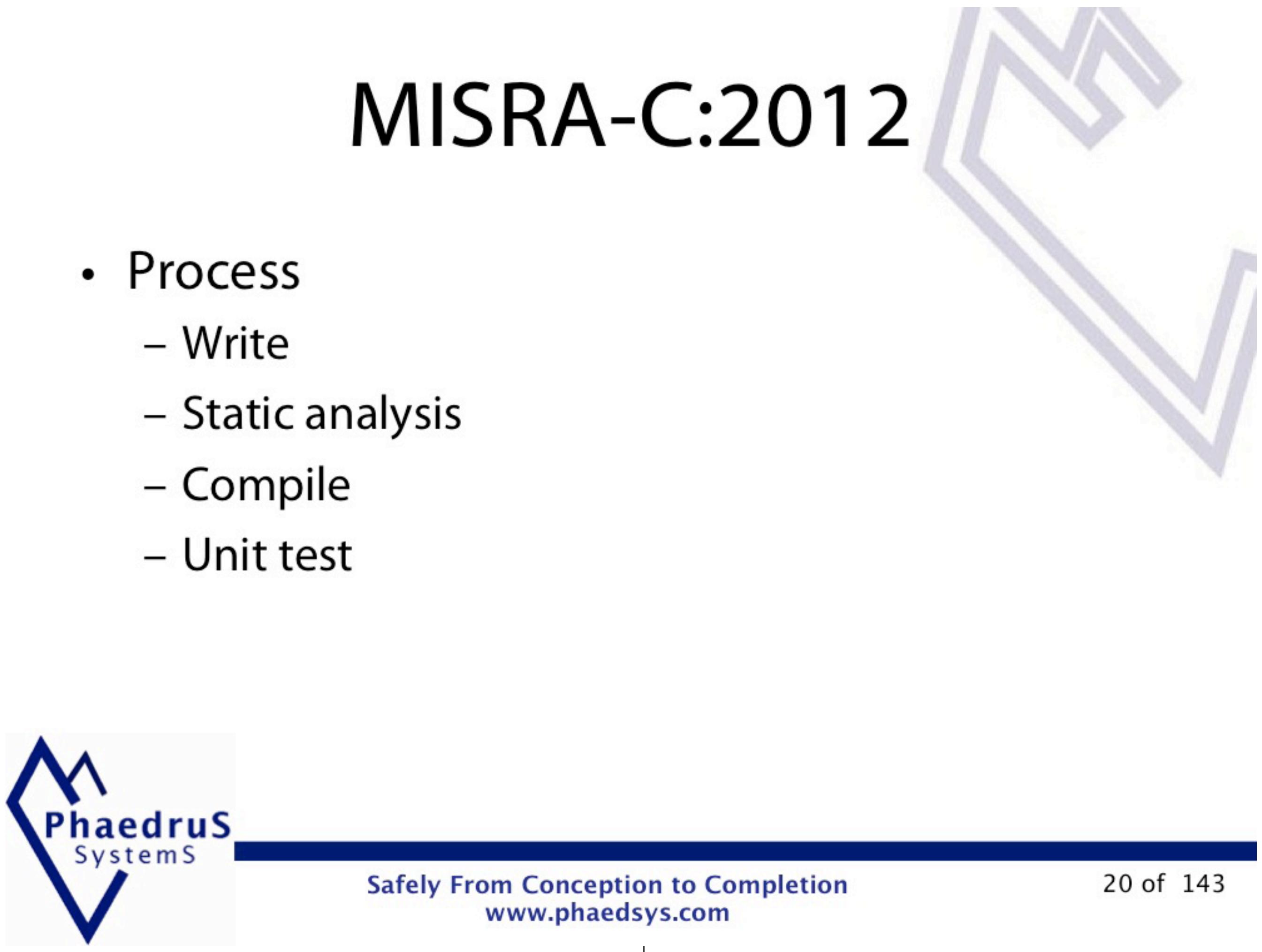
Static Analysis warns about things a compiler most certainly will not and should not warn about.

Most programmers are nothing like as clever as they think they are and the one sitting next to you certainly isn't! You really don't want to have to debug their code "later" when the deadlines are looking tight and you want to get home for the match.

A compiler does NOT warn of the very many legal but highly dangerous constructs that often (if not always) cause problems later.

Whilst many, if not all, of the dangerous features are useful once in a blue moon, you don't want, or need most of them most of the time. More to the point you need to know when they have been used, usually accidentally or for the wrong reasons. Hence static analysis and MISRA-C

Many studies show static analysis works and ***SAVES TIME AND MONEY.*** Most static analysis tools pay for themselves the first time they are run by finding simple bugs that, if they escaped into the wild, could cost several times the cost of the static analyser. That includes non-safety related projects.



MISRA-C:2012

- Process
  - Write
  - Static analysis
  - Compile
  - Unit test

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

20 of 143

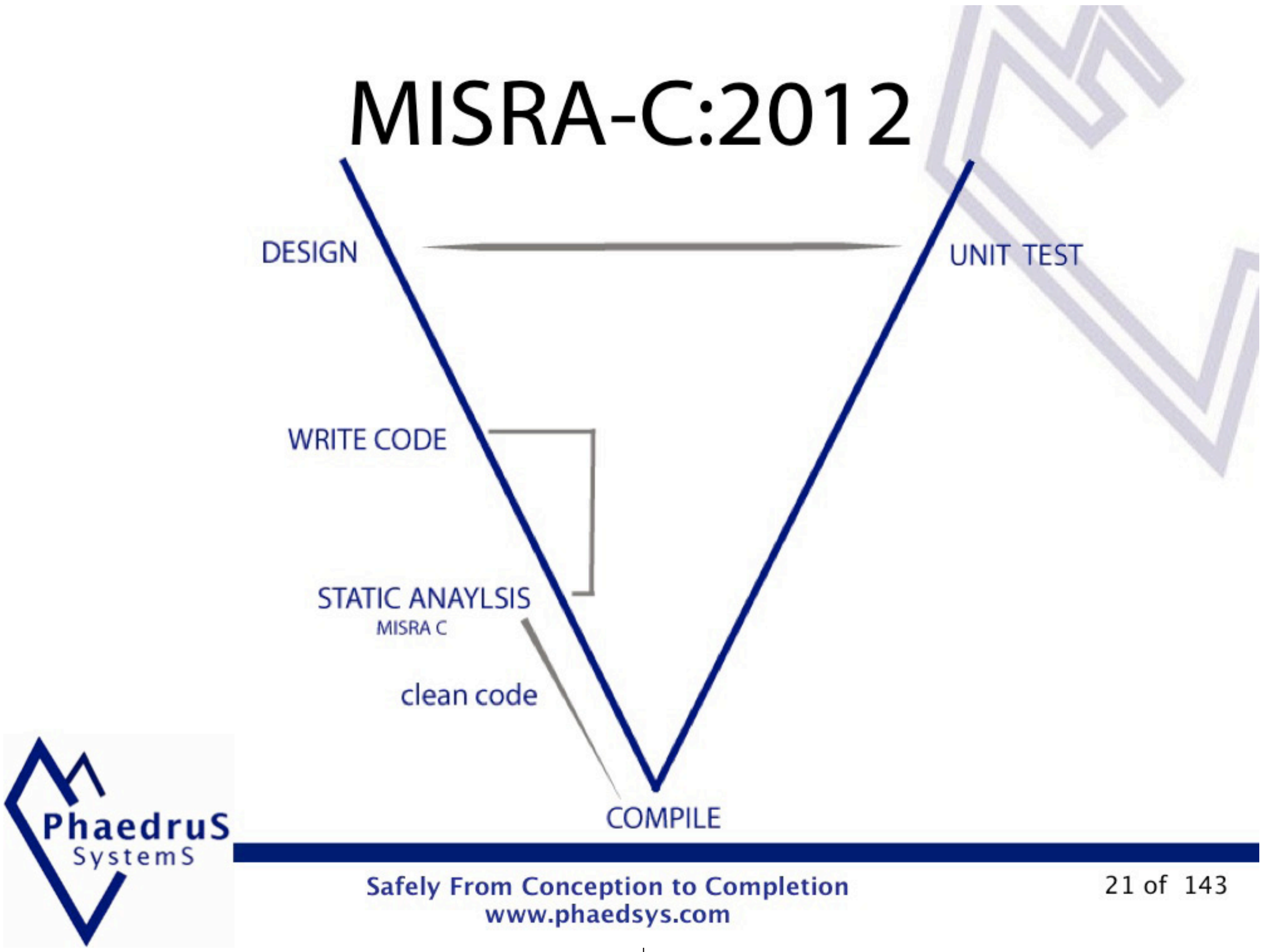
There is no point in compiling syntactically legal code that is dangerous. All you have compiled is code that may have bugs and may or may not behave correctly.

Therefore run the static analysis tool OFTEN as you write code. This should also check for MISRA-C violations. When you have corrected the errors and warnings (or deviated them and adjusted the static analysis configuration) take the next step.

You should set the compiler to the highest warning level and then compile. There should be no compile errors or warnings. You may still get linker errors and warnings. These linker errors will be outside the scope of the static analysis.

If there are compiler errors or warnings this means that, no matter how theoretically correct the source code, the compiler that is actually producing the binary has a problem and the binary is suspect. Therefore ALL compiler errors and warnings must be investigated and resolved.

At this point the code is as correct as it can be statically. The question is, "Did you program it to the specification?" So move on to Unit Testing, which is where you test to



This is the lower half of the V model in more detail.

Design specs should give unit test cases, which go to unit test. This means that you should have the test cases BEFORE you write the source code.

The next step is to write code to the Design Specifications and the coding standard. Then, as described in the previous slide, run the static analysis and MISRA-C checking.

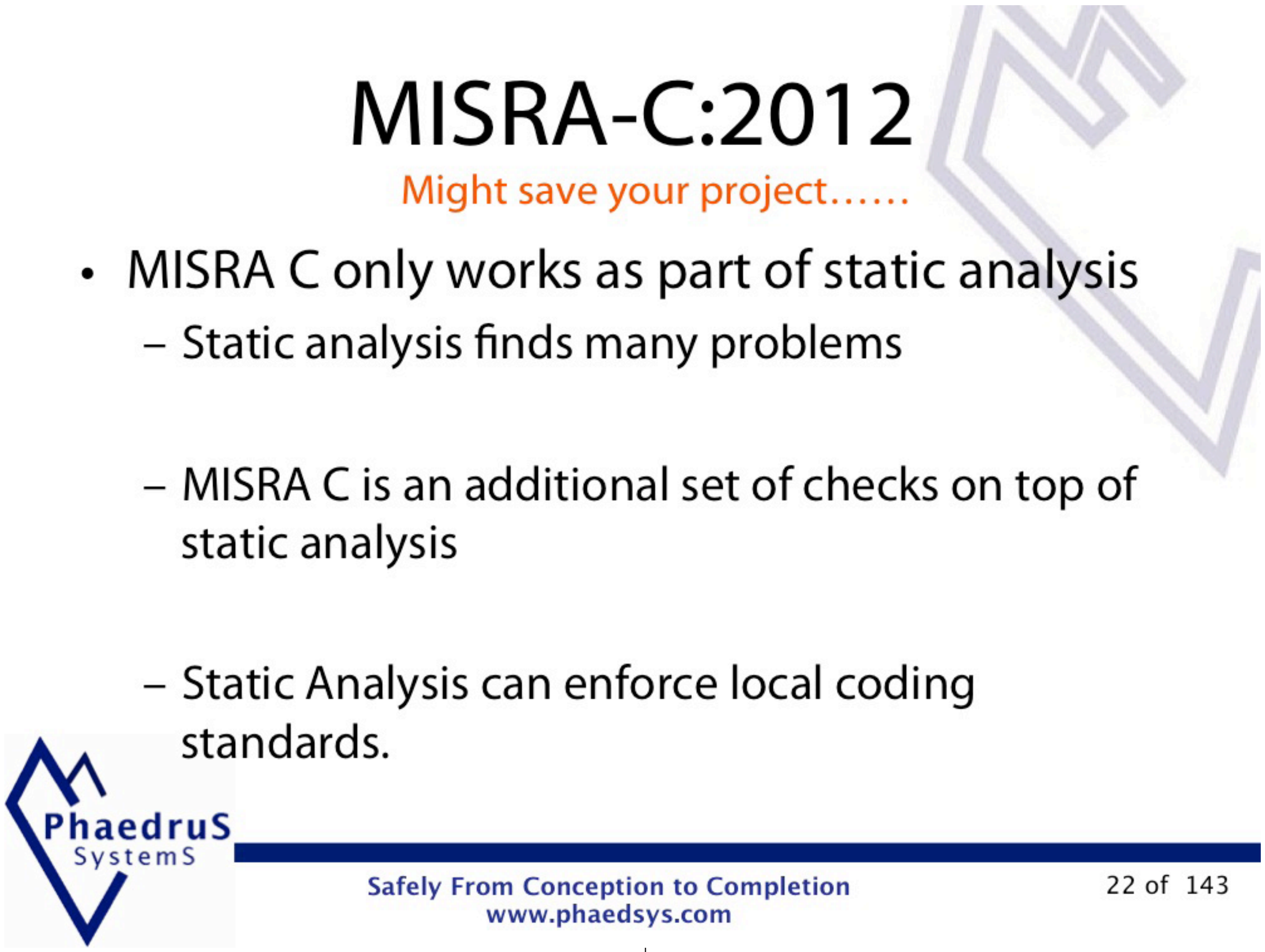
There is NO POINT in running MISRA-C checking unless you have run static analysis. MISRA-C restricts a subset of the C language. It is only a small part of static analysis which typically finds 1500 problems whereas MISRA-C has 143 additional Rules.

This will give you clean code.

Now compile the code, permitting no compiler errors other than architecture specific or compiler specific errors. Remove (actually resolve is a better word) all compiler errors and warnings. The solutions will depend on the problem.

NOTE. If you unit test before static analysis, you proved nothing. When you statically test you will find bugs. Fixing the code renders all the unit tests invalid. So unit test before static analysis it is a complete waste of time.

The same goes for compiling the code before static analysis. When you have done the static analysis you will have to compile again anyway.



MISRA-C:2012

Might save your project……

- MISRA C only works as part of static analysis
  - Static analysis finds many problems
  - MISRA C is an additional set of checks on top of static analysis
  - Static Analysis can enforce local coding standards.

PhaedruS SystemS

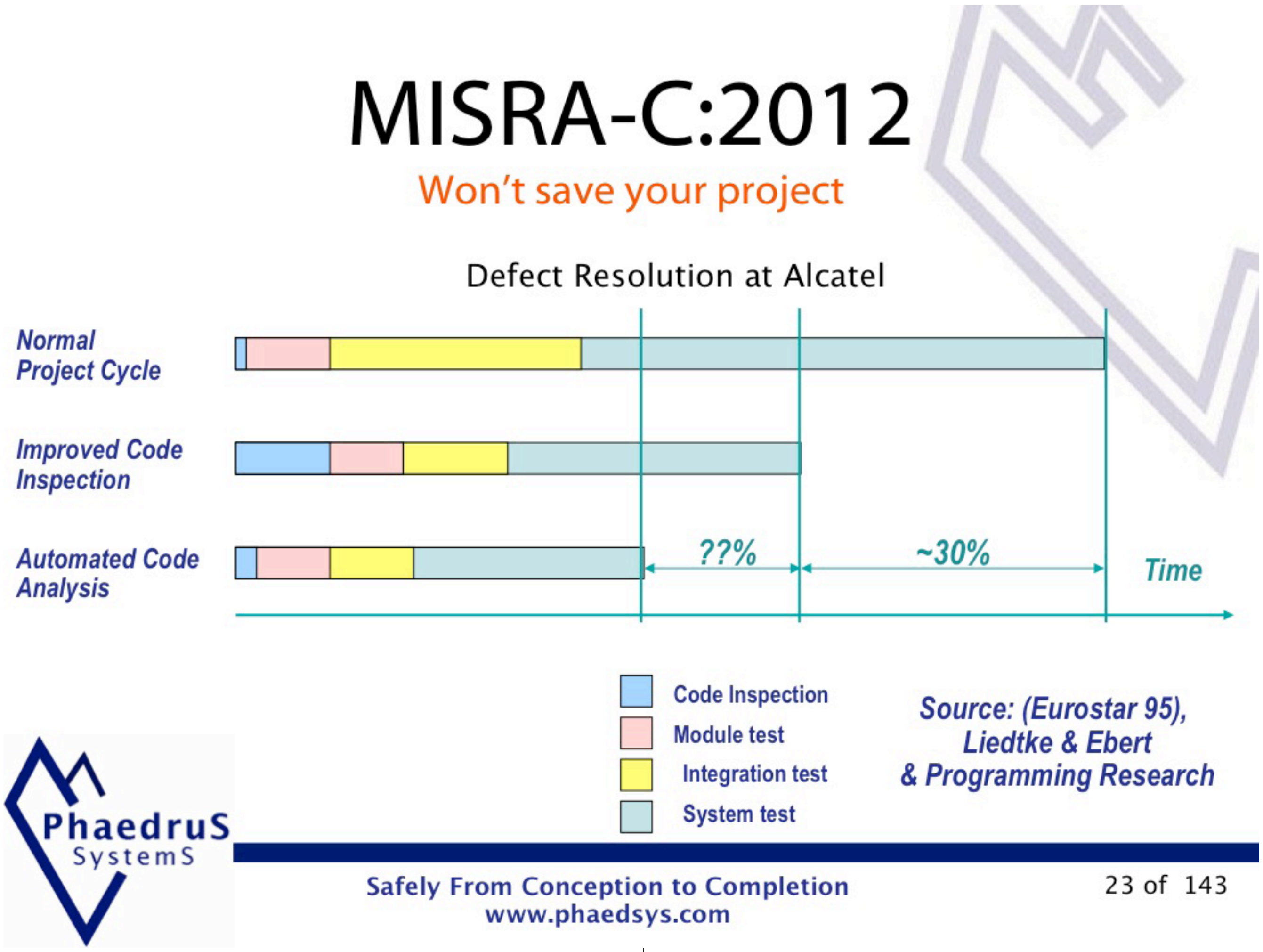
Safely From Conception to Completion
www.phaedsys.com

22 of 143

It is worth reiterating that MISRA-C should be used as part of static analysis. You also need a company coding standard and the static analyser can enforce this too. The analysis tool should be able to enforce both.

Static analysis is the only cost effective, time effective and reliable way to enforce a coding standard and MISRA-C. The big point is that MISRA-C is an addition to static analysis. You need static analysis for C. If you are not doing it there is no point in bothering with MISRA-C None at all.

Not doing static analysis (and then adding MISRA-C to it) is commercial suicide.



MISRA-C:2012

Won't save your project

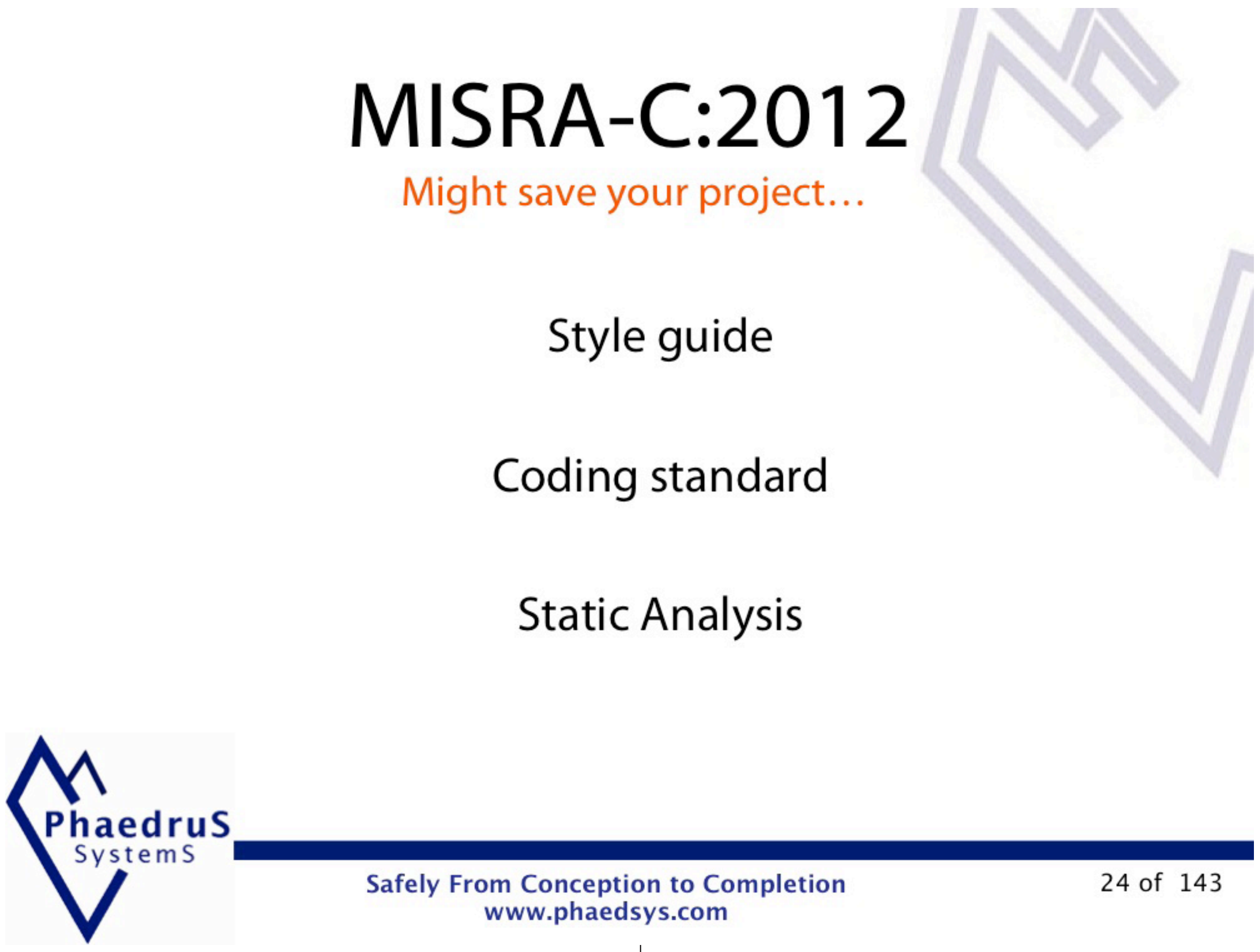
Safely From Conception to Completion
www.phaedsys.com

23 of 143

There are very many studies that show how important static analysis is. This one, from 1995 showed, that Improved/Formal code inspection shortened a project by 30 %. This assumes a style guide so all the code is uniform.

Automating this phase with static analysis saves even more time. With static analysis enforcing MISRA-C automatically, the code reviews are looking only at the code structure. You can "see the wood from the trees." Because the code review is no longer "bug hunting" in the code, reviews are far more productive and generally less fraught.

The same can be said for unit testing. With this phase also automated you are testing more (against the specifications) and bug hunting less. This gives a far higher degree of confidence in the system and again saves a lot of time. Automated unit testing systems also normally pay for themselves very quickly.



MISRA-C:2012

Might save your project…

Style guide

Coding standard

Static Analysis

PhaedruS SystemS

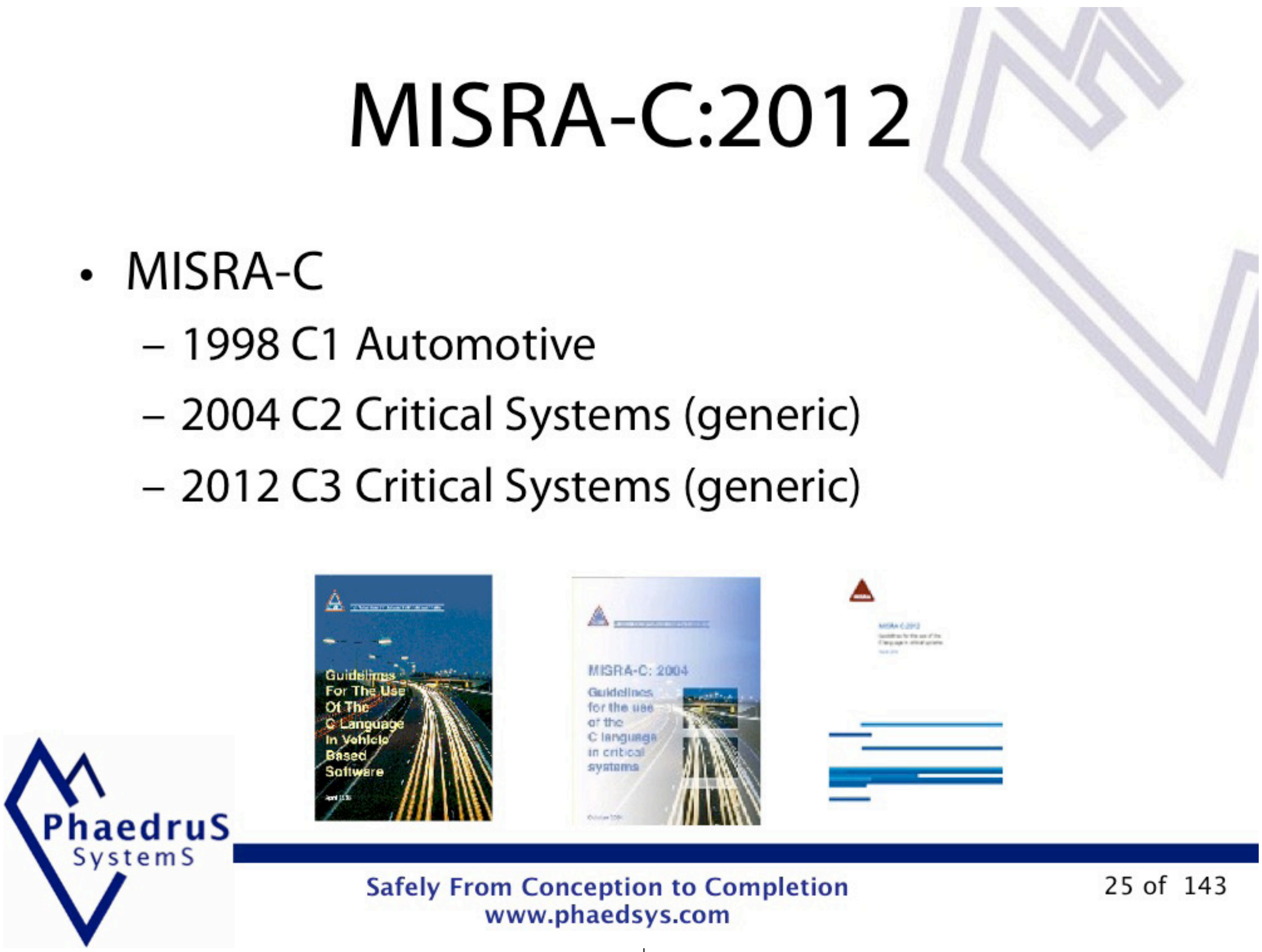
Safely From Conception to Completion
www.phaedsys.com

24 of 143

Style covered, static analysis covered. Any one not using a style guide and static analysis may as well not bother with MISRA-C and coding standard

MISRA-C should be part of the company coding standard and enforced at the static analysis phase. The company coding standard will normally be a mix of coding style and language subset. Which style it is does not matter much, as long as it is consistent. Style guides have improved over the 35 years since K&R's was written. A lot has been learnt since then and the language has changed. So I would not recommend the K&R style personally.

Now we move on to MISRA-C.



MISRA-C:2012

- MISRA-C
  - 1998 C1 Automotive
  - 2004 C2 Critical Systems (generic)
  - 2012 C3 Critical Systems (generic)

Guidelines For The Use Of The C Language In Vehicle Based Software
MISRA-C: 2004 Guidelines for the use of the C language in critical systems

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

25 of 143

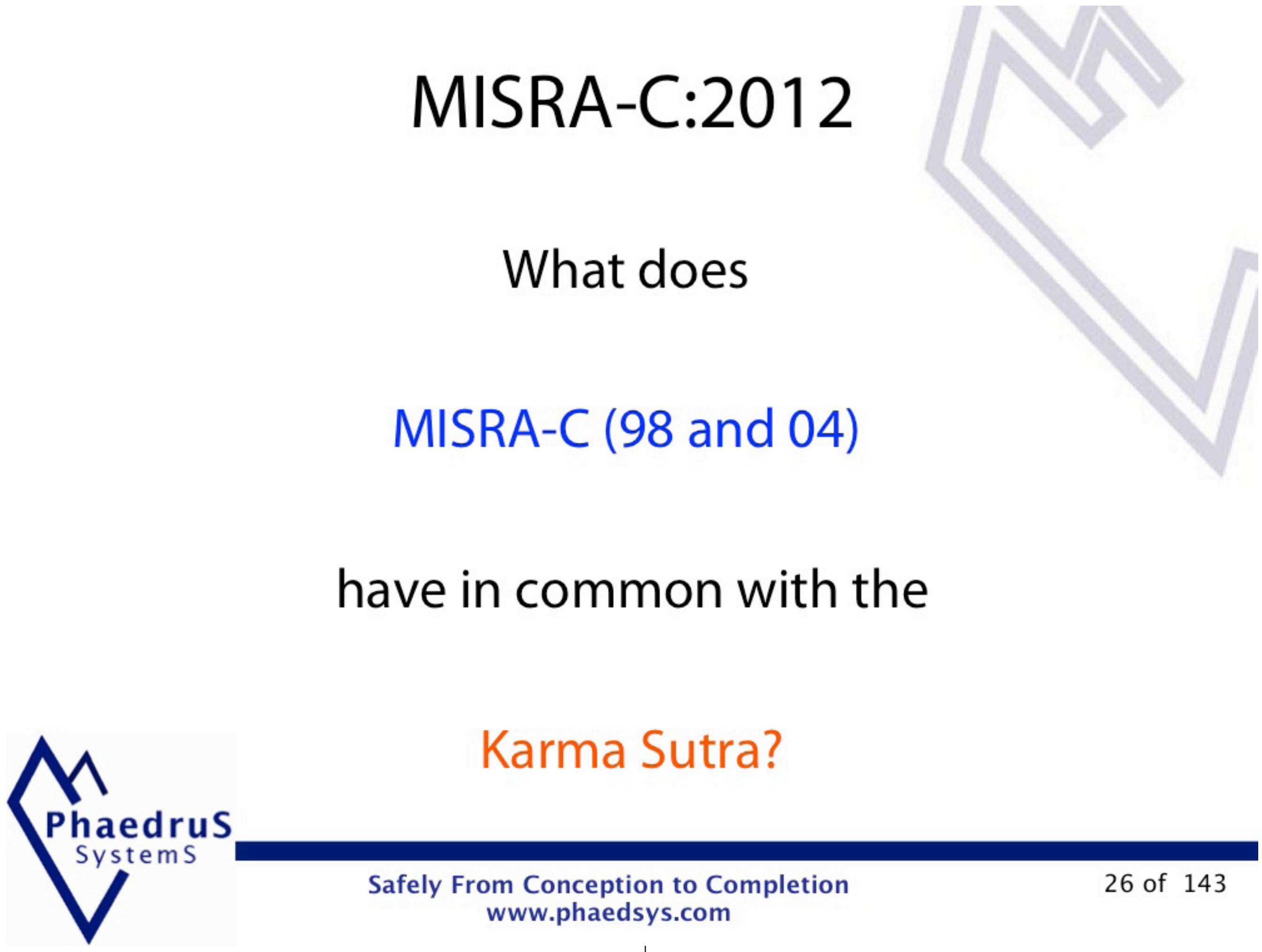
The first MISRA-C in 1998 was the last in a series of MISRA-Guidelines on software development for the UK Automotive industry. The guidelines were a local forerunner of 61508/26262.

MISRA-C was almost an afterthought as "report 9" However Programming Research (who had a hand in MISRA-C 1998) LDRA and Phaedrus Systems CTO pushed MISRA-C to a wider audience. See my initial review written in February 1998 http://www.phaedsys.demon.co.uk/chris/misra-c/misrac.htm As the review says this guide is ***suitable for all embedded C*** not just automotive.

In 2001 a MISRA-C working group was formed to start the next version. The team for C2 was only 50% automotive and the title changed to "Critical Systems" to reflect this and also to reflect the fact that in the 5 years since the release of C1 it become used way outside the automotive industry.

This trend continued apace and MISRA-C:2004 has been used as the basis of several major non-automotive coding standards, including the US Joint Strike Fighter (JSF) C++ coding standard and numerous company coding standards.

It is a fairly safe bet to say, that there probably isn't a single industry that is not using MISRA-C somewhere. This includes nuclear, rail, medical, marine, oil and gas, aerospace, defence etc. In fact the biggest group on the MISRA-C:2012 team is defence and aerospace: automotive now makes up only 10% of the C3 and current MISRA-C team. With a team of 10 from diverse industries, we have over 250 years experience, mainly on high integrity and safety related systems in the field. The tool vendors on the team see hundreds if not thousands of diverse projects. Add to this we also get feedback from the MISRA forum..



MISRA-C:2012

What does

MISRA-C (98 and 04)

have in common with the

Karma Sutra?

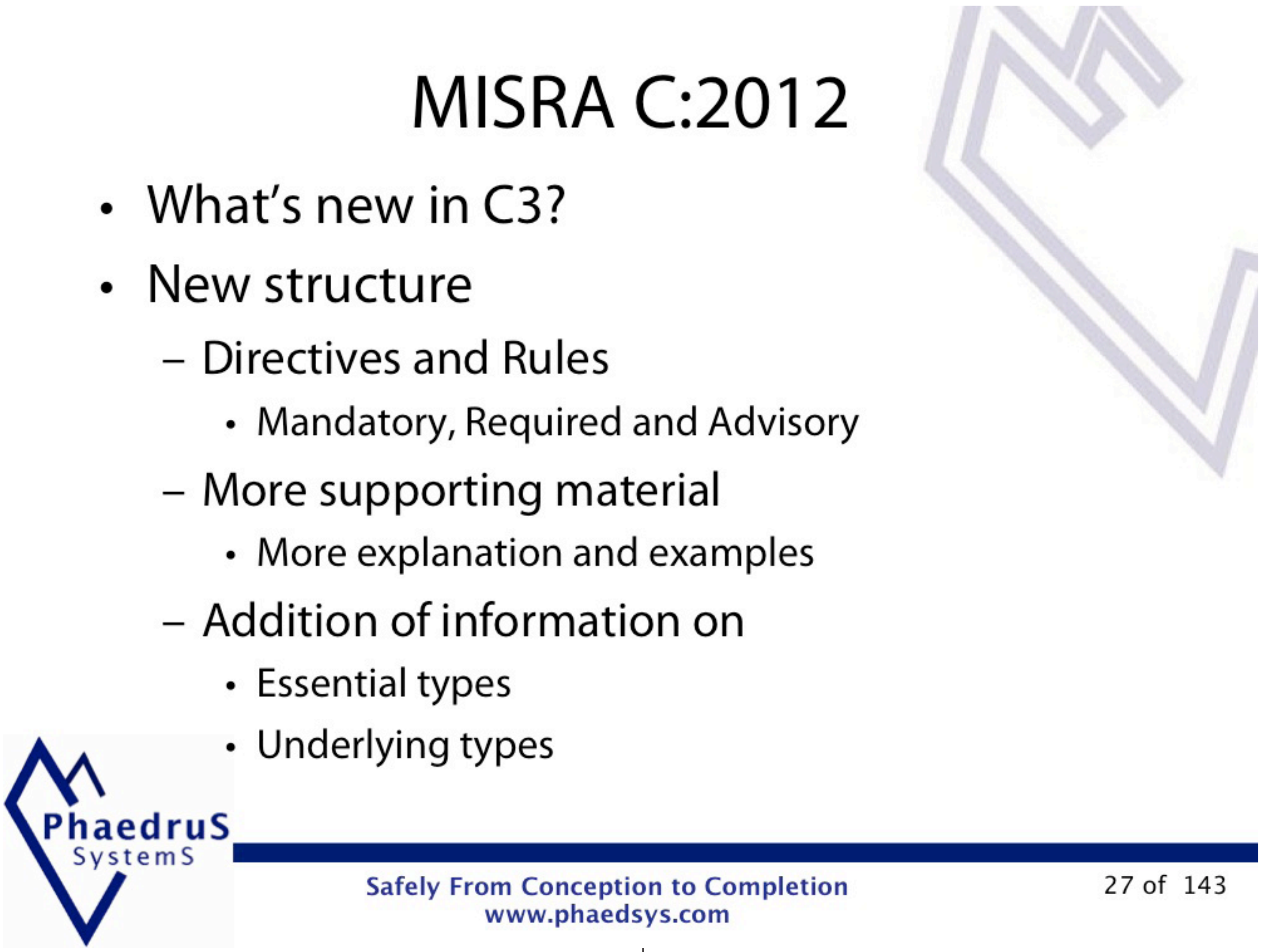
Safely From Conception to Completion
www.phaedsys.com

26 of 143

This slide usually raises some smiles because the majority of people don't realise that like MISRA-C:2004 The Karma Sutra has 7 parts. The only part most people know of is the one part that that contains sex. (part 2) They have never heard of the other 6 parts. In the same way with MISRA-C: 98 and 04 most people only seem to read the rules and skip the other 6 chapters. These other chapters are the most important parts! They tell you how to implement the rules.

MISRA C:2012 has 9 (NINE) chapters and many appendices. However it is now even more true that you need to read all the parts of MISRA-C except the rules before you start to do anything with MISRA-C.

To implement MISRA C you need to understand it, so you need to read ALL of it. Not just the rules. This time around, MISRA-C:2012, ***you WILL have to read all of it or you will not be able to claim any sort of MISRA compliance.***



MISRA C:2012

- What's new in C3?
- New structure
  - Directives and Rules
    - Mandatory, Required and Advisory
  - More supporting material
    - More explanation and examples
  - Addition of information on
    - Essential types
    - Underlying types

PhaedruS SystemS

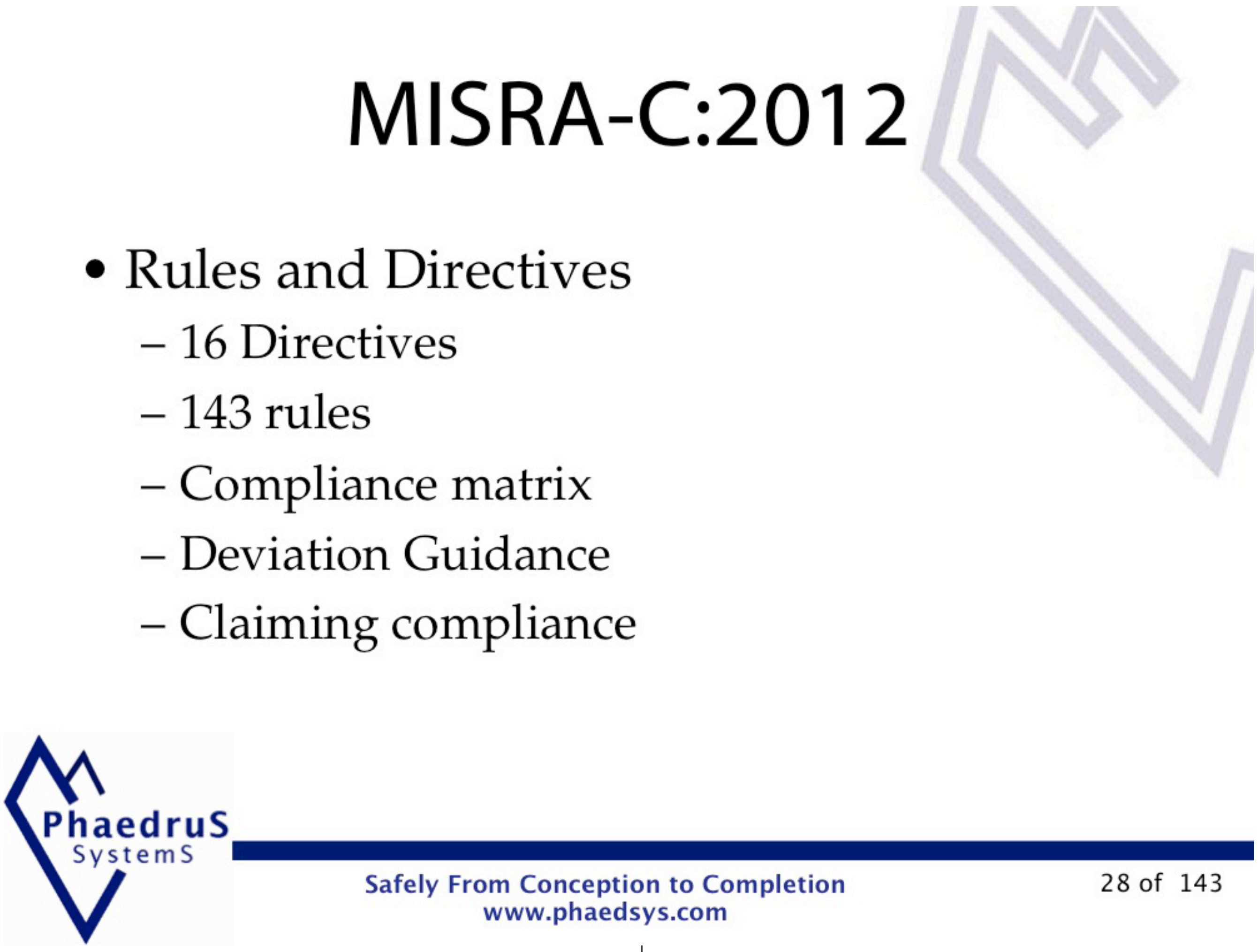
Safely From Conception to Completion
www.phaedsys.com

27 of 143

What's new on MISRA-C:2012? A new structure: We now have Rules and Directives. The directives have the same weight as the rules but are rules that can not be directly taken from the source code alone. Also there is an additional MANDATORY category.

We have put in a lot more supporting material which is why MISRA-C is almost twice the size of the last version with only a 10% increase in the number of rules.

There is a lot of material that we always intended to release as an additional document for MISRA-C:2004, discussing essential and underlying types. This is integer promotion on steroids! This section was written by Paul Burden, one of the longest serving MISRA team members and THE authority on underlying and essential types. Paul works for Programming Research and you should contact them for more information.



MISRA-C:2012

- Rules and Directives
  - 16 Directives
  - 143 rules
  - Compliance matrix
  - Deviation Guidance
  - Claiming compliance

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

28 of 143

Rules and Directives. The reason for splitting them is that rules apply only to source code and most (about 80%) can be statically determined.

The Directives have the same weight as the rules but cover matters that you can not discern just from the source code. For example, rule 4.2 requires that all usage of assembly language should be documented. The directives are in no way lower or lesser than the rules.

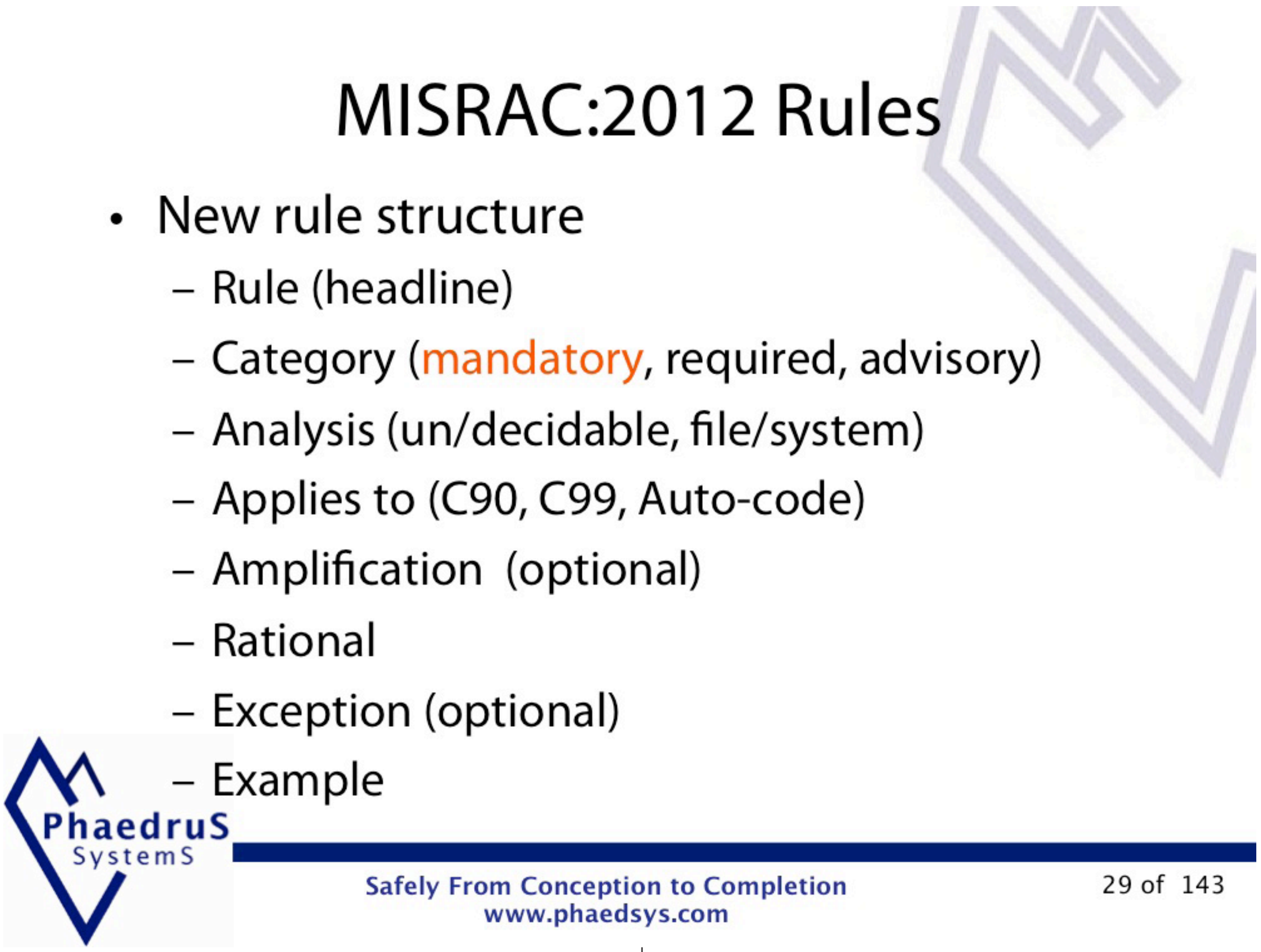
There is still the compliance matrix which is one of those things that has been a part of the guidelines since MISRA-C:1998 and has been ignored as it is in the first 6 chapters However in order to do MISRA-C at almost any level bar the informal you need a compliance matrix.

Deviation guidance is new BUT requires you have a compliance matrix, which you will all have had if you have done any MISRA-C enforcement....

We included some guidance on Deviation procedures as many people asked for them.

There is more on Claiming MISRA-C compliance. which also explains why you need a Deviation document that requires a compliance Matrix……

There are no short cuts in MISRA-C:2012



MISRAC:2012 Rules

- New rule structure
  - Rule (headline)
  - Category (mandatory, required, advisory)
  - Analysis (un/decidable, file/system)
  - Applies to (C90, C99, Auto-code)
  - Amplification (optional)
  - Rational
  - Exception (optional)
  - Example

Safely From Conception to Completion
www.phaedsys.com

29 of 143

MISRA-C 2012 has a modified rule structure. The Headline Rules are now short and succinct, without all the exclusions. (They were in danger of becoming 5 line rules reading like the titles of academic papers.) These are followed by the category for the rule including the new "Mandatory" category.

The Analysis is new. This says, if the rule is theoretically decidable or not, whether in a single file, or across the project. About 80% are "decidable" in theory, so any tool claiming 100% MISRA testing is being, to put it kindly, enthusiastic. It should be possible to reach nearly 90% decidable, if certain rules are not deviated. We (the MISRA-C team) are currently (July 2013) looking doing some work to identify how this can be achieved and the rules concerned.

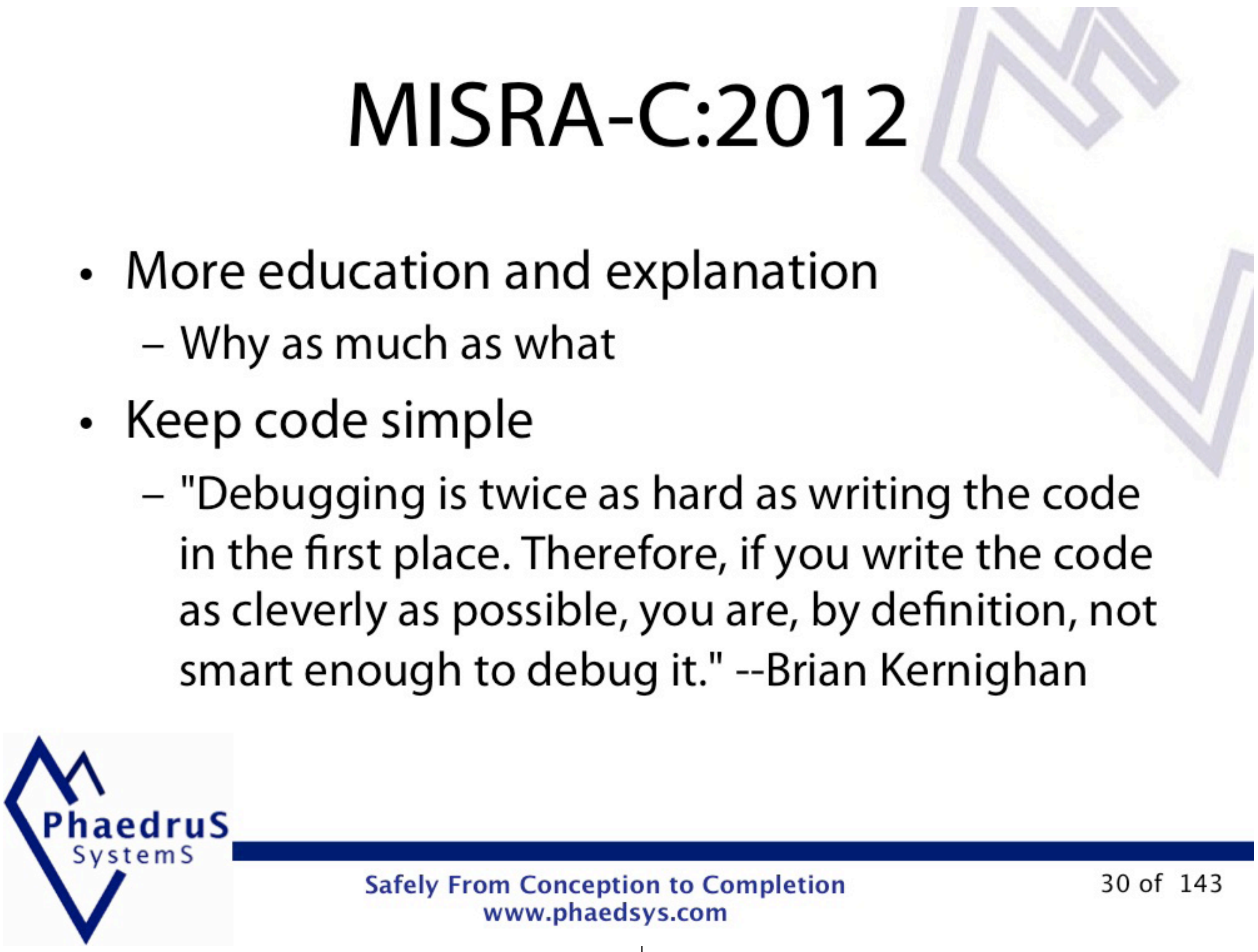
The "Applies To" is important. C90 compilers behave differently to C99 compilers. Do you know which C you use? (And did you know "ANSI-C" died in 1990 when it was superseded by ISO9899:1990…) This also marks the rules that applied to the MISRA-AC autocode documents.

The [optional] Amplification is the additional explanation of the rule so we could have the short headline rules. Which why now, more than ever, reading just the headline rule is pointless.

The Rationale explains the thinking and the reasons for the rule. This removes any excuses for following the letter of the headline rule but ignoring the spirit of the rule. **There are no excuses any more.**

The [optional] Exceptions again help make a simplified headline rule and the exceptions listed do not need to be deviated. These examples are illustrative and not exhaustive. There will be other positives and negatives we have not mentioned

Finally every rule has several illustrative positive and negative examples.



MISRA-C:2012

- More education and explanation
  - Why as much as what
- Keep code simple
  - "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it." --Brian Kernighan

Safely From Conception to Completion
www.phaedsys.com

30 of 143

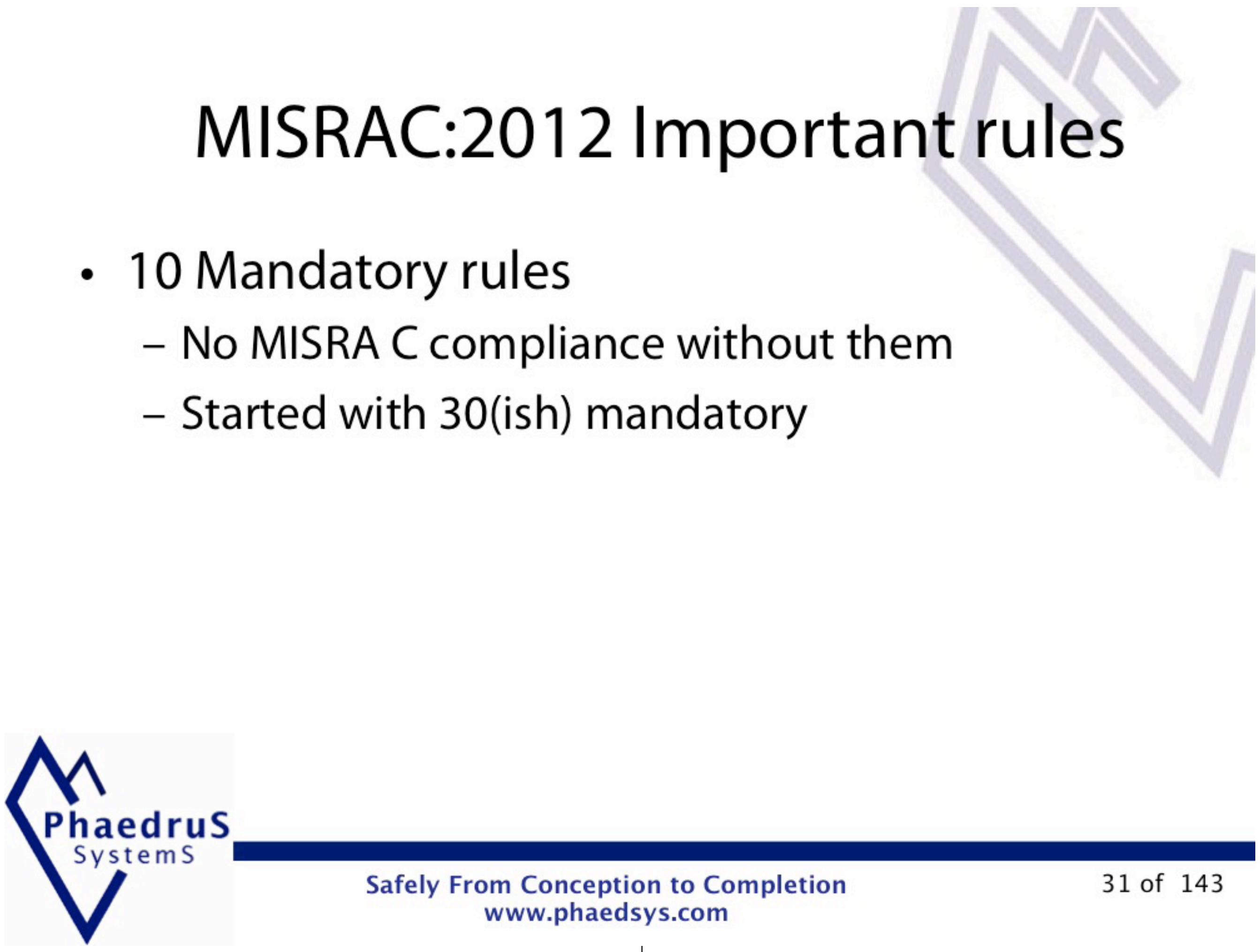
As you can see there is a lot more description and explanation in MISRA-C:2012. If the truth be told over the 8 years of development (or some 80 man years) we made notes in our development system as to why we were changing and adapting the rules for MISRA-C:2004

This is why you need to read all of MISRA-C and not just the rules, let alone just the headline rules. The overriding lesson is: Do not try to be clever! Do not try and "beat the rule"

As Brian Kernighan said, "keep it simple" If you are that clever you should be in debug and maintenance otherwise no one will be able to debug your code. Over the years I have found this to be true.

Programmers "being clever"generally have difficult-to-read code and more time is spent de-cyphering the code than hunting the bug.

I have on several occasions unravelled some clever code to find a bug and in doing so, found other un-reported bugs as well. When the code was rewritten in a simple way it was obvious what the code did and there was nowhere for the bugs to hide



MISRAC:2012 Important rules

- 10 Mandatory rules
  - No MISRA C compliance without them
  - Started with 30(ish) mandatory

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

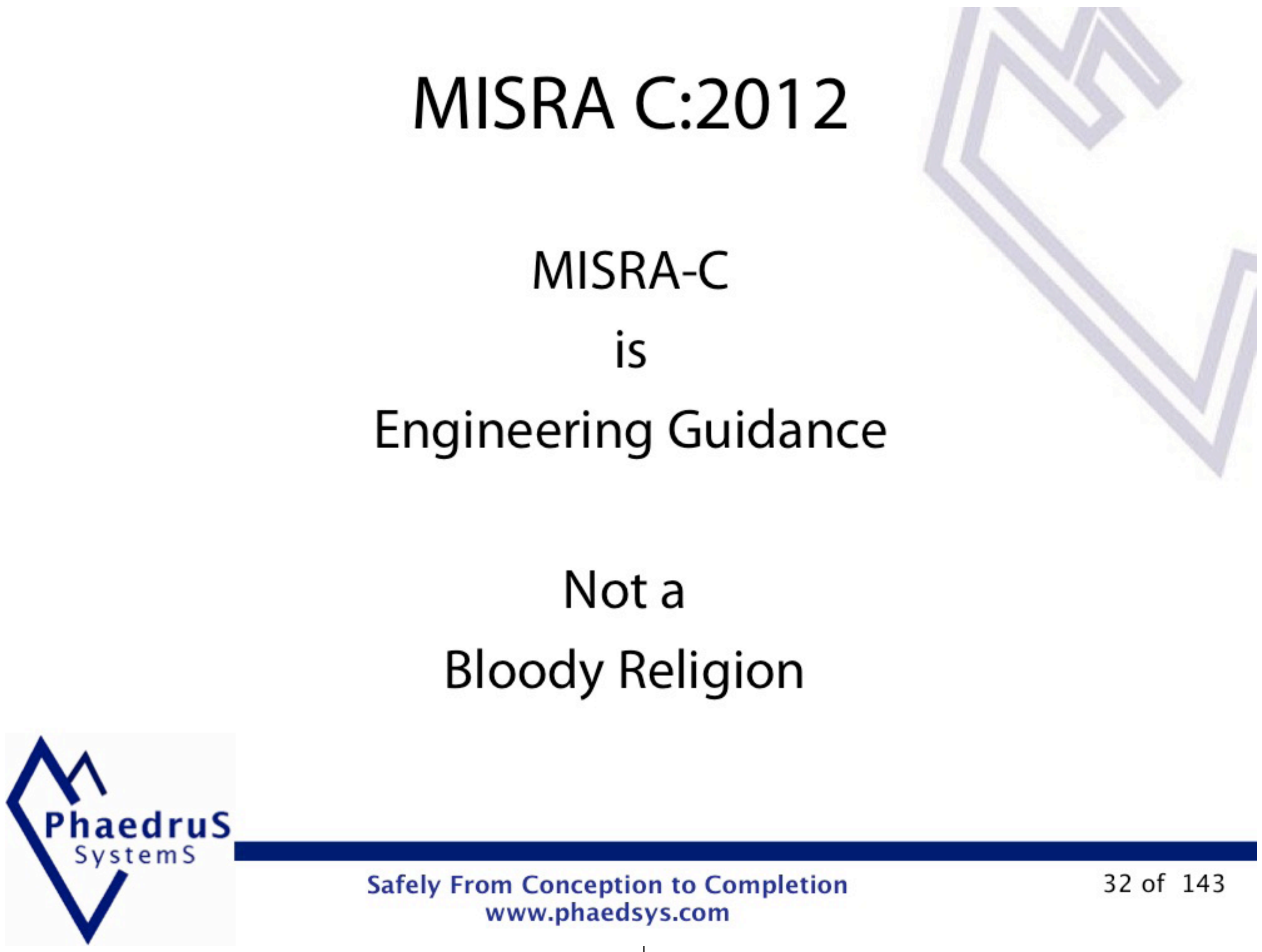
31 of 143

There are 10 Mandatory rules. Only 10 out of 159 Rules and Directives.

We actually started with about 30 mandatory rules but people kept finding legitimate reasons to deviate them. In the end we had 10…

So only 7% of MISRA-C:2012 rules will hold true 100% of the time. This is why there are deviations… There are not THAT many things that are universally true for C because of architecture, extensions and restrictions, also the nature of the project.

Unions, for example, we banned but expect those doing communication streams to deviate them. This is why you need a Compliance Matrix and Deviation Guide.



MISRA C:2012

MISRA-C

is

Engineering Guidance

Not a

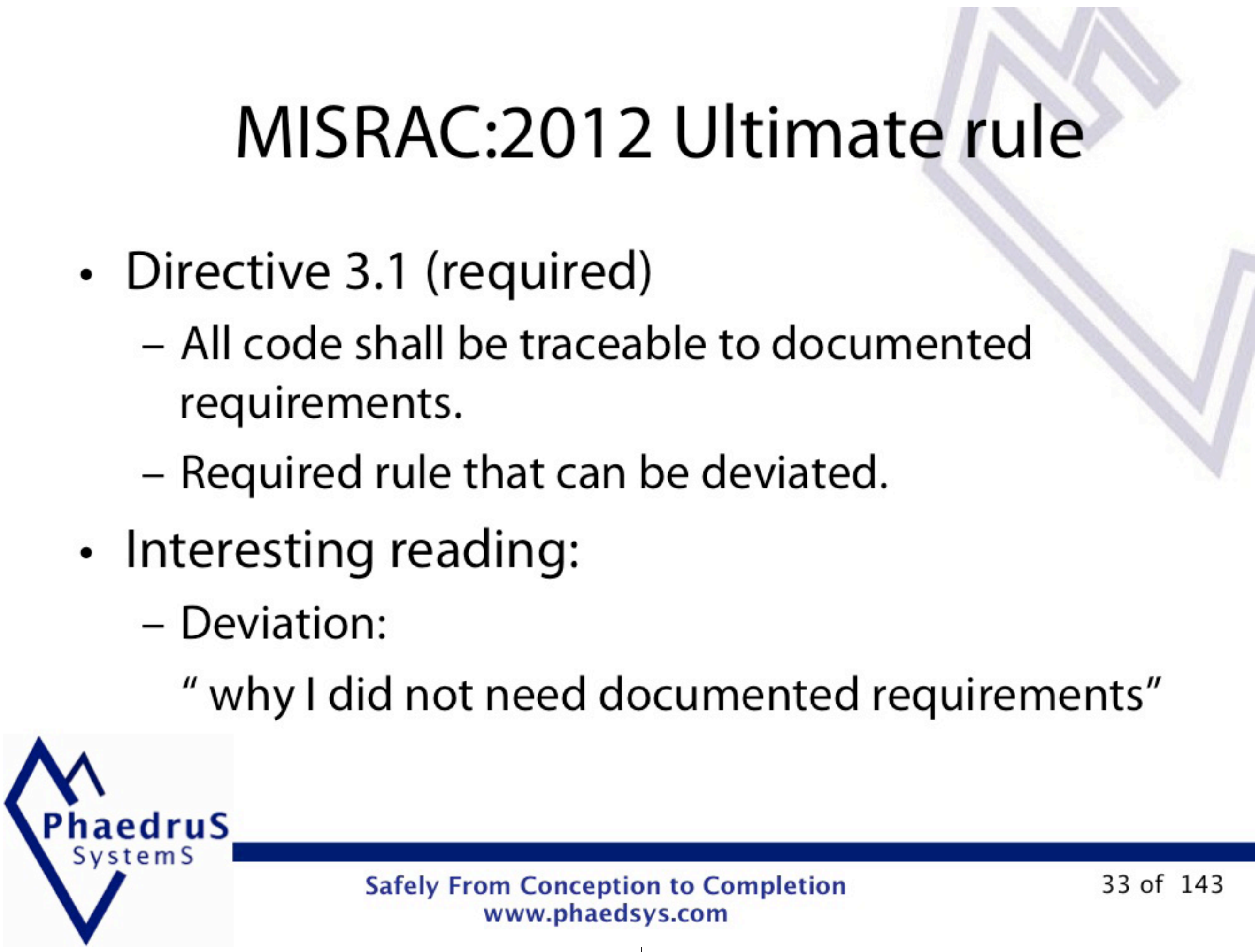
Bloody Religion

Safely From Conception to Completion
www.phaedsys.com

32 of 143

More explanation and 8 other chapters to read…… ***MISRA-C:2012 is NOT just a tick box***. You have to read, understand and apply "sensibly"

It is not a religion to be followed blindly. It is Engineering Guidance but you have to be able to justify your decisions.



MISRAC:2012 Ultimate rule

- Directive 3.1 (required)
  - All code shall be traceable to documented requirements.
  - Required rule that can be deviated.
- Interesting reading:
  - Deviation:

    " why I did not need documented requirements"

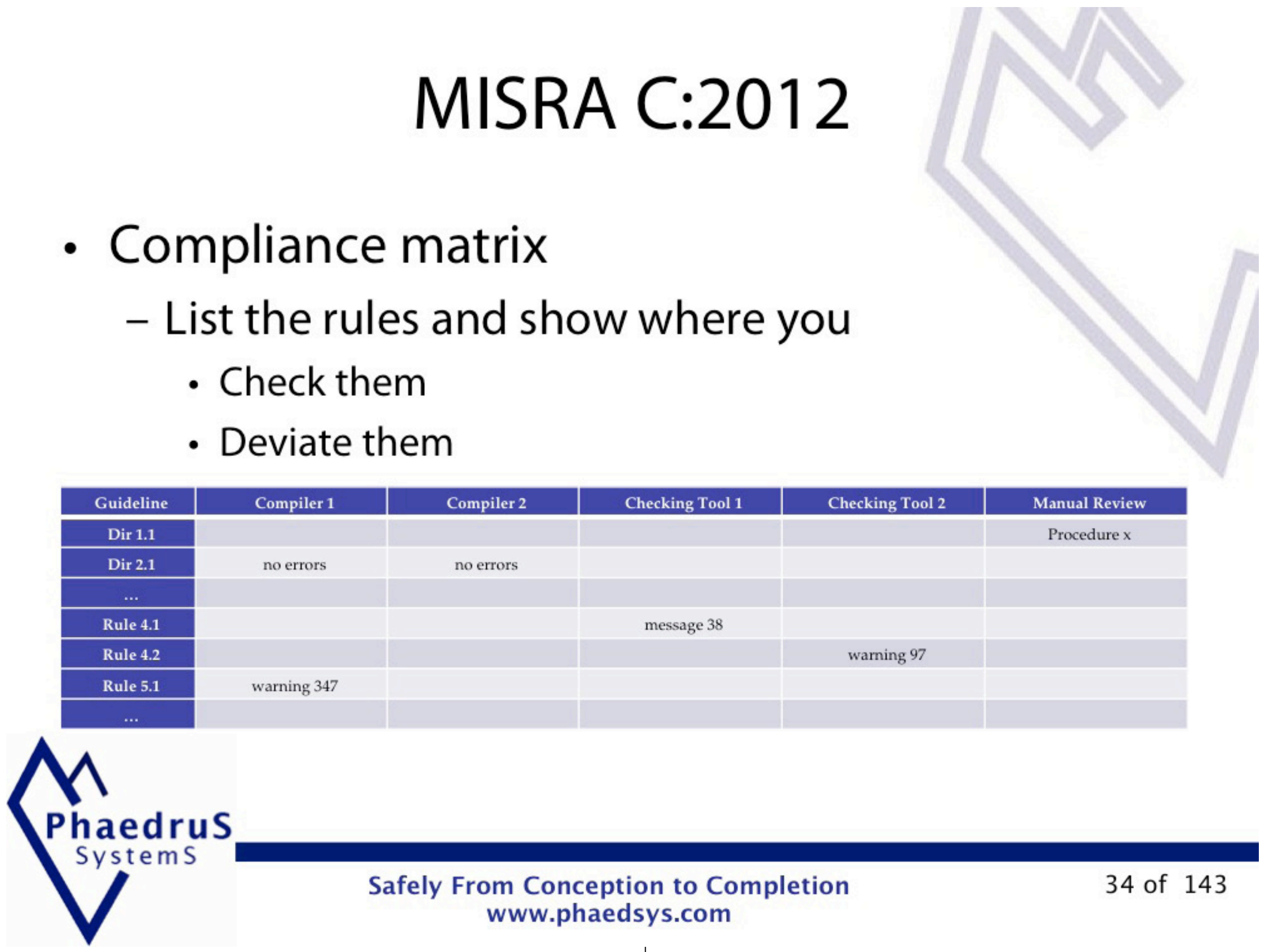
Safely From Conception to Completion
www.phaedsys.com

33 of 143

The ultimate MISRA-C:2012 rule is Directive 3.1 (Required) As it is required you can deviate this directive BUT in order to do so you have to show why you do not need requirements. And why you do not have to trace them to the code.

So, if you can come up with a good reason why you wrote code you don't have proper requirements to write…

This rule is a game changer as it puts the responsibility back on to the people enforcing MISRA-C in the company. If they don't deviate MISRA-C then you need full sets of requirements and traceability traceability to the code. This means that you can't start writing code unless the requirements are complete (and some one has signed for them). Or, in the other case, if some one has taken responsibility and signed for the deviation then you can start writing code without full requirements. That should get a few people thinking!



MISRA C:2012

- Compliance matrix
  - List the rules and show where you
    - Check them
    - Deviate them

Guideline	Compiler 1	Compiler 2	Checking Tool 1	Checking Tool 2	Manual Review
Dir 1.1					Procedure x
Dir 2.1	no errors	no errors			
...					
Rule 4.1			message 38		
Rule 4.2				warning 97	
Rule 5.1	warning 347				
...					

PhaedruS SystemS

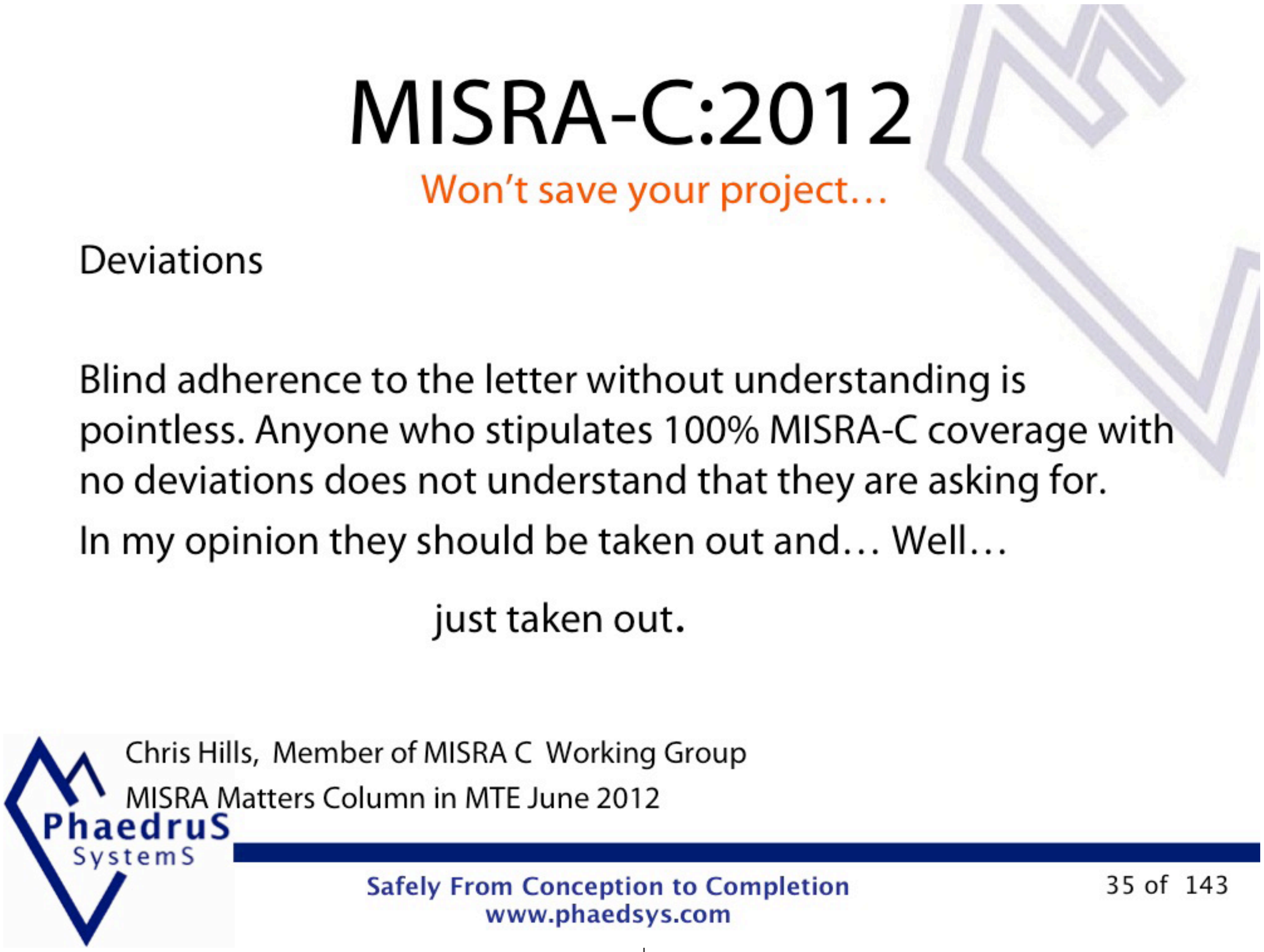
Safely From Conception to Completion
www.phaedsys.com

34 of 143

This has not changed since MISRA-C1 in June 1998, apart from the colours in the table! You list ALL the rules and where they are checked. Some will appear in more than one column. There will be an overlap between the compiler and the static analyser, but the main MISRA-C checker should be the static analyser.

There will also be a manual review column. Static analysis and MISRA-C does not negate the need for a formal code review and indeed some of the directives require that one be done.

A Compliance matrix is easy to do in a spread sheet, word processor or even formal Requirements Management software. It does not matter how you do one but YOU WILL NEED ONE. There will be at least one entry against every rule including the deviated rules. You will need a column for deviations. This is where you put the reference for a deviation.



MISRA-C:2012

Won't save your project…

Deviations

Blind adherence to the letter without understanding is pointless. Anyone who stipulates 100% MISRA-C coverage with no deviations does not understand that they are asking for.

In my opinion they should be taken out and… Well…

just taken out.

Chris Hills, Member of MISRA C Working Group

MISRA Matters Column in MTE June 2012

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

35 of 143

Deviations are something we are asked about a lot. There are two classes of discussion. Firstly, "How do I deviate," which I will cover next. Secondly how to meet the requirement for "100% MISRA-C no deviations[TICK]." This is usually…. actually, always, imposed by people who don't understand what MISRA-C is or how to implement it.

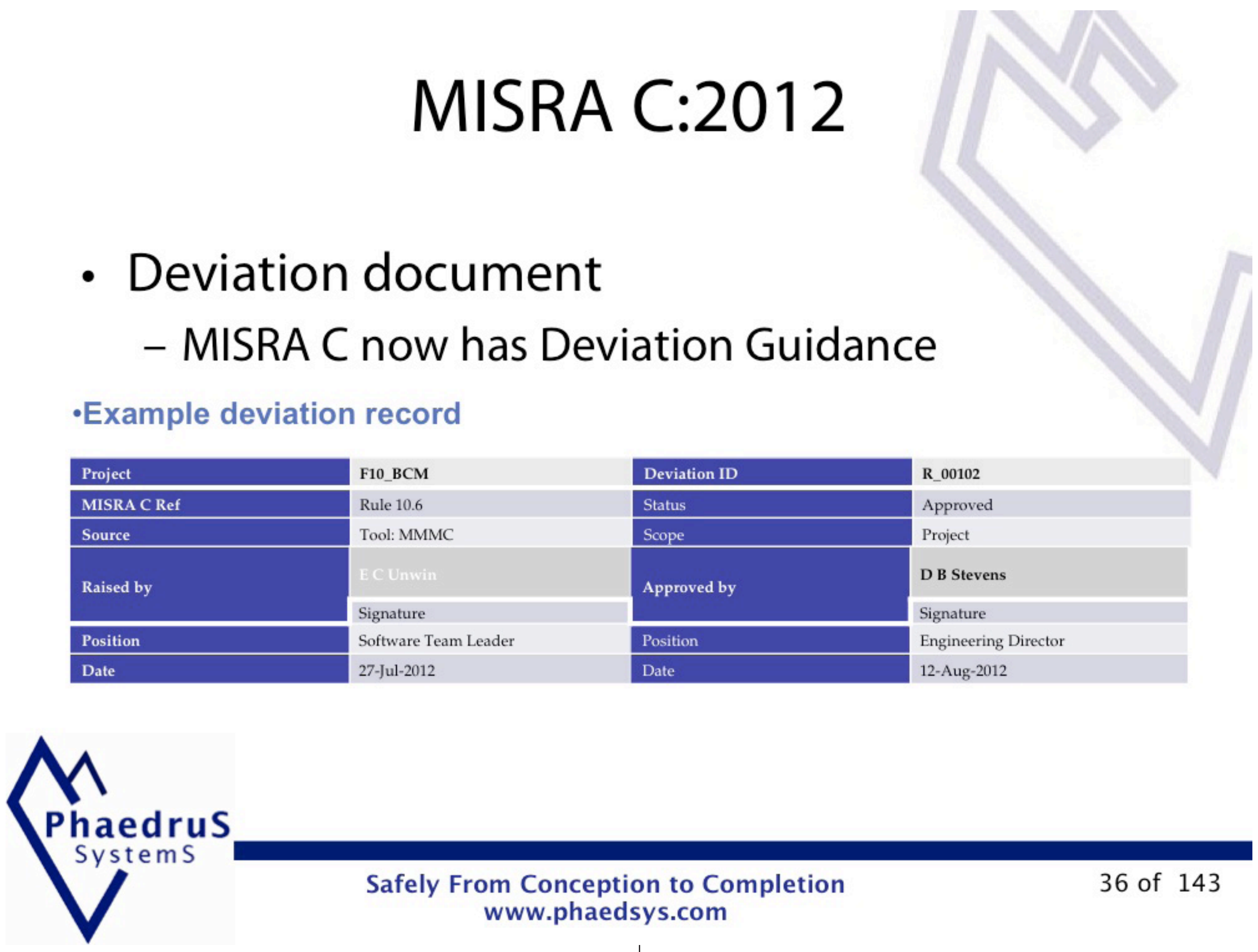
As mentioned there are only 6% of the MISRA-C rules that are Mandatory.: That is rules that are applicable 100% of the time. Therefore we hope that 99.9999% of MISRA-C users will deviate the appropriate rules.

This is one of the places where MISRA-C can be counter productive. When, for example, some manager demands 100% compliance without realising he is dangerously handicapping the project. The team fight with the standard and resorts to all sorts of time consuming, and in some cases dangerous, tricks to get round the warnings from the code analysers.

They spend a lot of time getting hideous and less efficient code.

A4 size Copies of this slide are available signed for your manager's office wall!

Deviations WILL be required.



MISRA C:2012

- Deviation document
  - MISRA C now has Deviation Guidance

•Example deviation record

Project	F10_BCM	Deviation ID	R_00102
MISRA C Ref	Rule 10.6	Status	Approved
Source	Tool: MMMC	Scope	Project
Raised by	E C Unwin	Approved by	D B Stevens
	Signature		Signature
Position	Software Team Leader	Position	Engineering Director
Date	27-Jul-2012	Date	12-Aug-2012

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

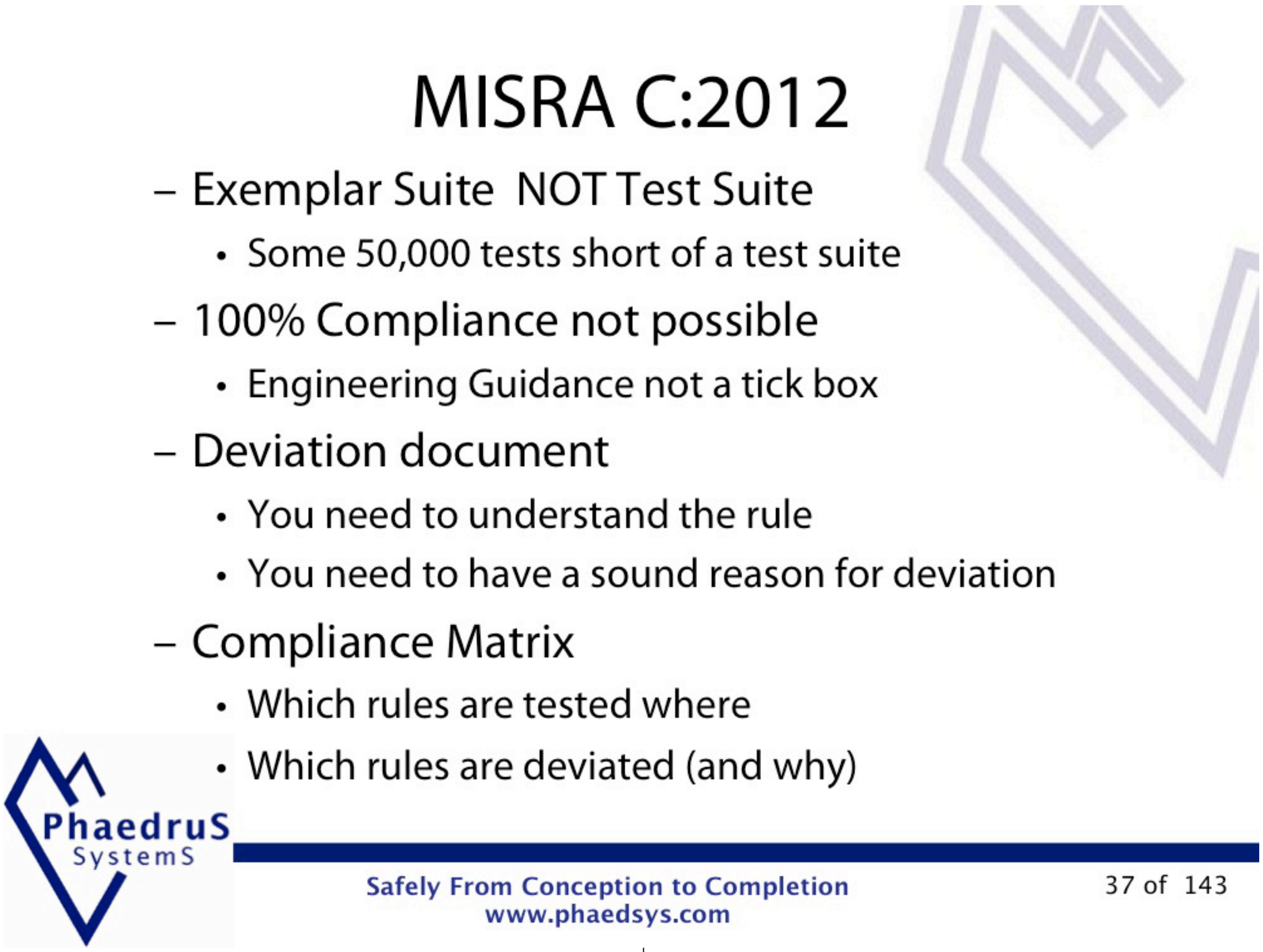
36 of 143

As a lot of people wanted an "approved" Deviation, we have written a couple of pages on deviation with an example above. However this is just a general example. Remember, we are offering Engineering Guidance not preaching a Religion, so modify the diagram and the suggested methods to fit your processes. It really does not matter what the form is or what looks like: it is the function that is important. The person who wrote the Deviation guidance worked for a large company in a specific industry, which shows. So use it as an editable template not a rigid form.

The overriding thing about a deviation is, does it make sense to some one else in 6 weeks time? I always say. "Go and ask your non-technical partner at home if the deviation makes sense to them. If you can't explain it simply in plain English then how will your deviation read to a board of enquiry or a jury in 6 years time?"

NOTE "neat", "cool", "Radical", "dude", "man", "wicked", "ace" and "bro" were not cool, radical, neat or wicked, man, the first time around in the 60's 70's, 80's etc and won't work now especially to a jury of 60 year old BCS or IET Charted Engineer Expert Witnesses.

So, seriously, simple plain English deviations for reasons that really stand up. Hopefully they will not need to stand up next to you in the dock in court!



MISRA C:2012

- Exemplar Suite NOT Test Suite
  - Some 50,000 tests short of a test suite
- 100% Compliance not possible
  - Engineering Guidance not a tick box
- Deviation document
  - You need to understand the rule
  - You need to have a sound reason for deviation
- Compliance Matrix
  - Which rules are tested where
  - Which rules are deviated (and why)

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

37 of 143

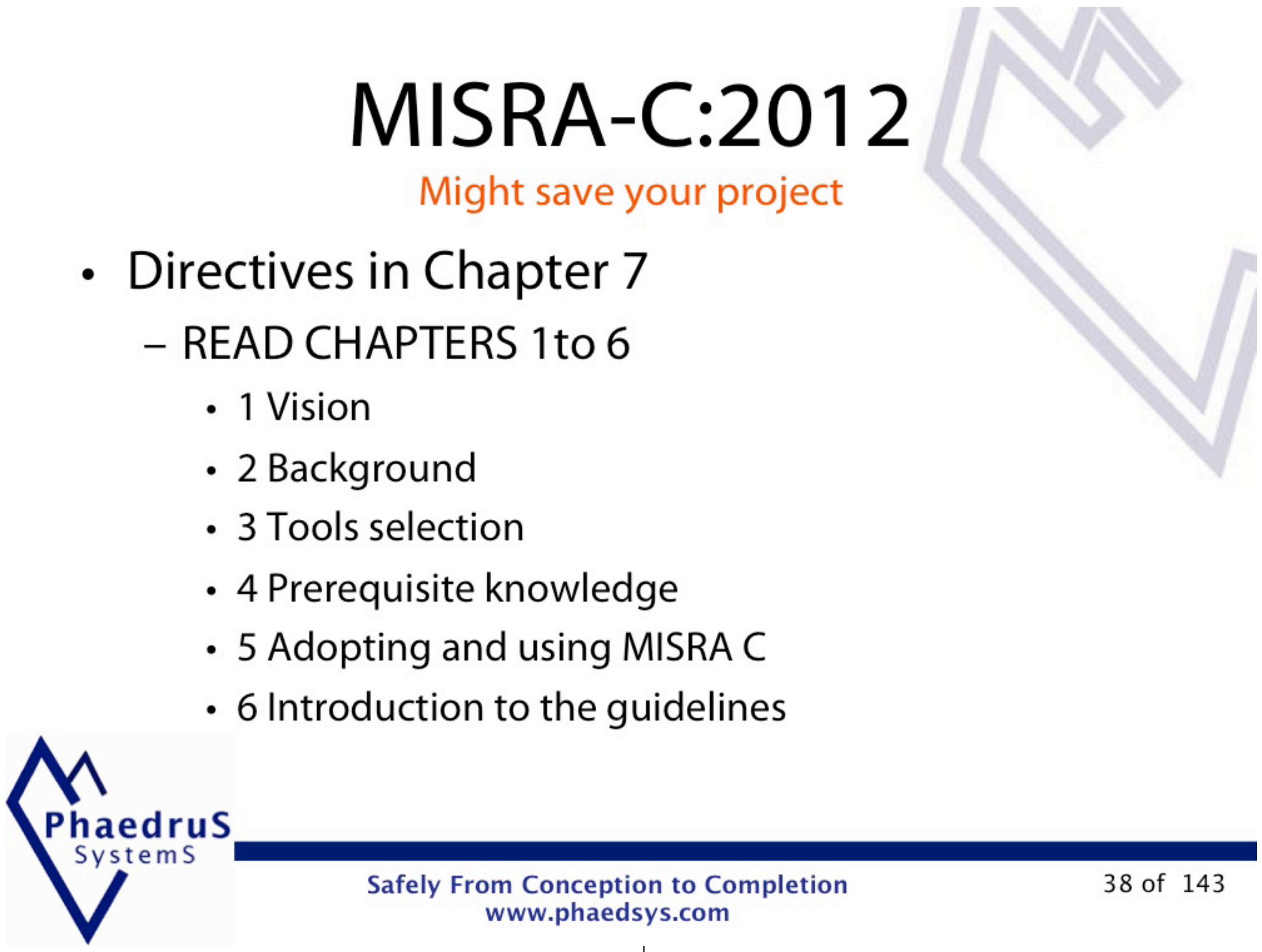
So in a round up of MISRA-C 2012 the exemplar suite will be made up of the examples pulled from the standard. They are NOT repeat ***NOT a test suite*** as we are some 50,000 tests short of a test suite.

The EXEMPLAR suite can not be used for testing tools against each other. The tests were there to help explain what we meant and for the team to test the rules in compilers and static analyzers to help formulate the rules..

For 99.9999% of MISRA-C users 100% MISRA-C compliance is not going to be a Good Thing. Sensible Engineering will require some deviations.

There fore you will need a deviation document. More to the point you will have to have read and understood the rule so you know why you are deviating and the deviation you write needs to make sense to some one outside your team a week after you wrote it.

In order to do MISRA-C you will need a compliance matrix to show where each rule is being checked and or deviated (and why) including a manual code review phase.



MISRA-C:2012

Might save your project

- Directives in Chapter 7
  - READ CHAPTERS 1to 6
    - 1 Vision
    - 2 Background
    - 3 Tools selection
    - 4 Prerequisite knowledge
    - 5 Adopting and using MISRA C
    - 6 Introduction to the guidelines

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

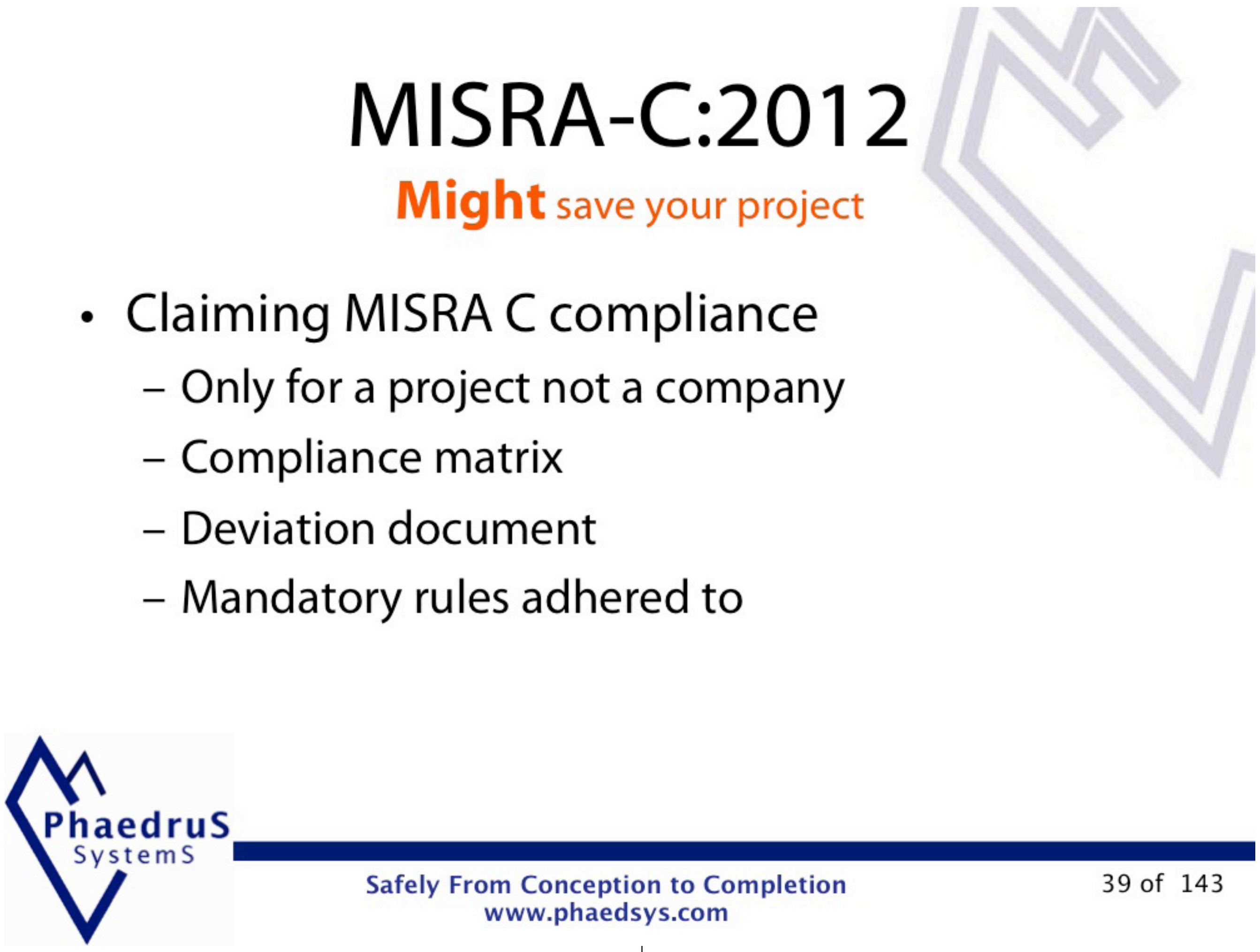
38 of 143

There is more to MISRA-C than the rules (and directives) there is a LOT more guidance on implementation of MISRA C.

The Vision and Background are nice but not essential - read them if you are bored one day.

Chapters 5 and 6 are essential and you must read these two parts if not any of the others.

Chapter 4 is also useful but I find Chapter 3 is somewhat misguided in places. (And I said so at the time.) I shall probably be writing a commentary on it at some point



MISRA-C:2012

Might save your project

- Claiming MISRA C compliance
  - Only for a project not a company
  - Compliance matrix
  - Deviation document
  - Mandatory rules adhered to

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

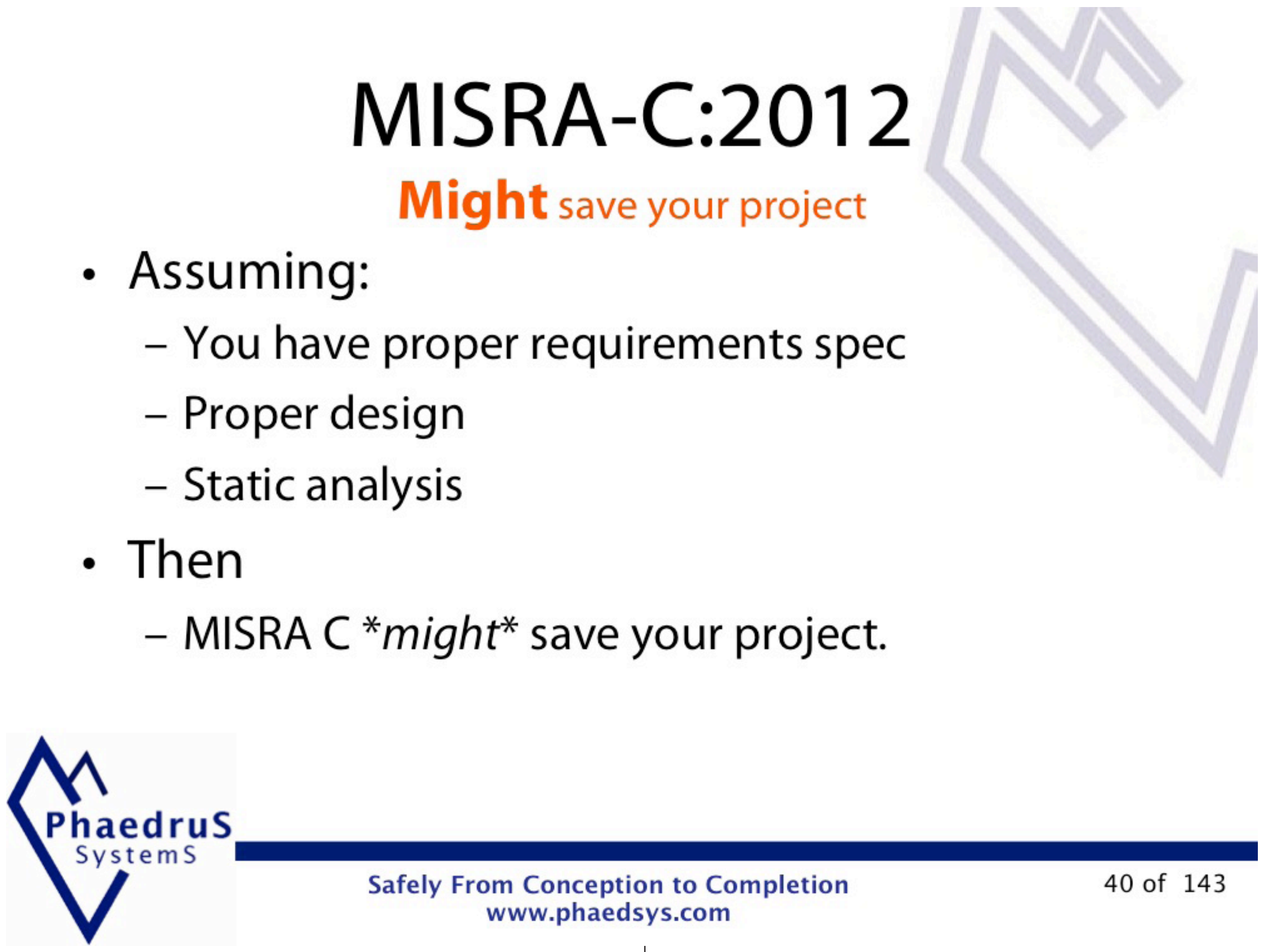
39 of 143

There are now notes on how to claim MISRA-C Compliance for a project. Not for a company, but only for an individual project.

You MUST have a completed compliance matrix and deviation documents. They must match each other and match the configuration of the MISRA-C checking tools: which WILL be a static analyzer. Practically speaking you can't claim MISRA-C compliance without one. Theoretically you could but it would take so much time and manpower that is it not a commercial option. You must of course adhere to the Mandatory rules (10 of them at the time of writing) if you are working to MISRA-C:2012. And do make it clear which MISRA-C you are working to. '98, '04 or '12.

Remember you may have to produce both the compliance document and the deviation documents to substantiate your claims so the deviations had better be sound. Of course you will need complete traceability between the requirements and the source code. That or a fascinating deviation why not!

At the end of each year Phaedrus Systems will give a prize to the best deviation of Directive 3.2 received in the year. misra@phaedsys.com



MISRA-C:2012

Might save your project

- Assuming:
  - You have proper requirements spec
  - Proper design
  - Static analysis
- Then
  - MISRA C **might** save your project.

PhaedruS SystemS

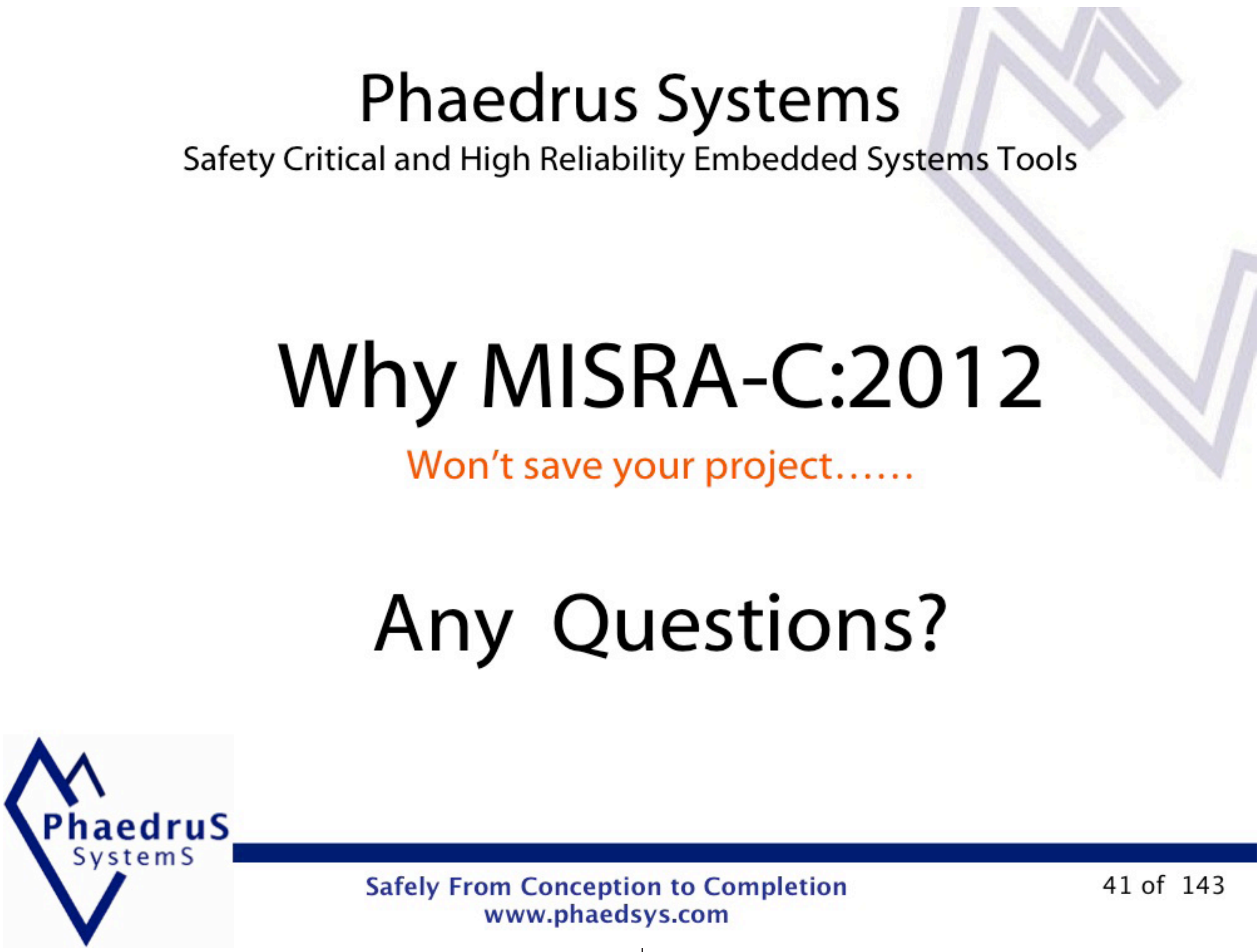
Safely From Conception to Completion
www.phaedsys.com

40 of 143

So MISRA-C or MISRA-C++ really only work well if you have a full requirements specification and a complete and reviewed design, so you know exactly what you are building, and then use MISRA-C as an addition to the static analysis phase.

Static analysis finds many problems and MISRA-C is an additional set of rules and restrictions.

What you can't do is impose MISRA-C at the end of a project.. You have to start writing MISRA-C code at the beginning. You can put MISRA-C on to legacy projects but a file at a time and be careful. Some systems only run because there are bugs in them. Cleaning up one module might close the door on another faulty module so that it now no longer works as it did or as it should.



Phaedrus Systems

Safety Critical and High Reliability Embedded Systems Tools

Why MISRA-C:2012

Won't save your project……

Any Questions?

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

41 of 143

So MISRA-C might save your project as part of a properly implemented system. On its own it is jut one more tool in the box. Like any other tool it can do more harm than good if misused. If you have any questions there are several places you can go for help:

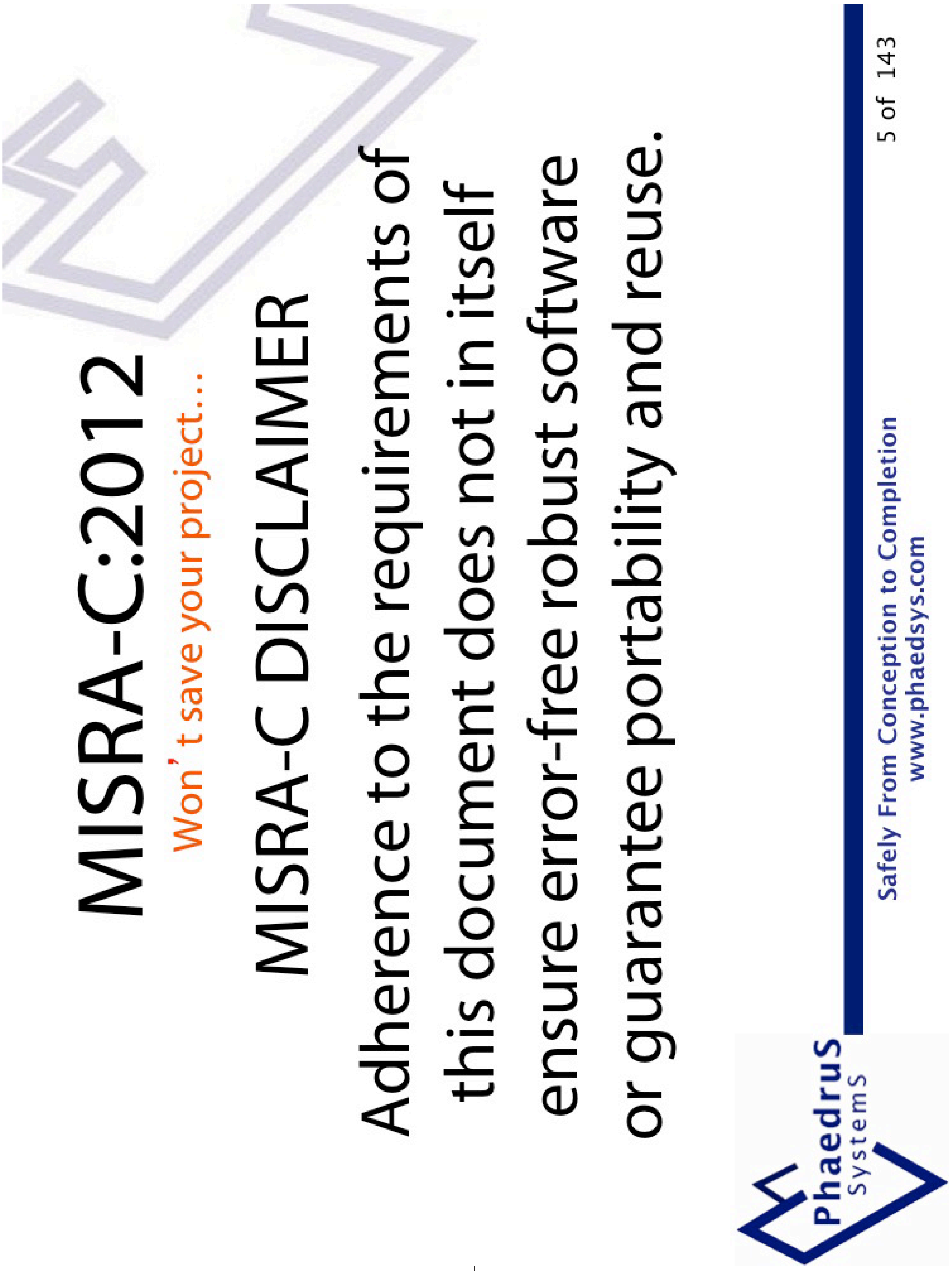
For authoritative and definitive statements from the MISRA-C Working Group got to www.misra-c.com/forum

For general discussion on MISRA C and C++ there is the LinkedIn forum "MISRA-C and C++" This is where most of the MISRA-C team hang out.

Otherwise for general MISRA-C information, static analysis, general SW Engineering and project control information contact Phaedrus Systems

MISRA@Phaedsys.com

www.phaedsys.com



MISRA-C:2012

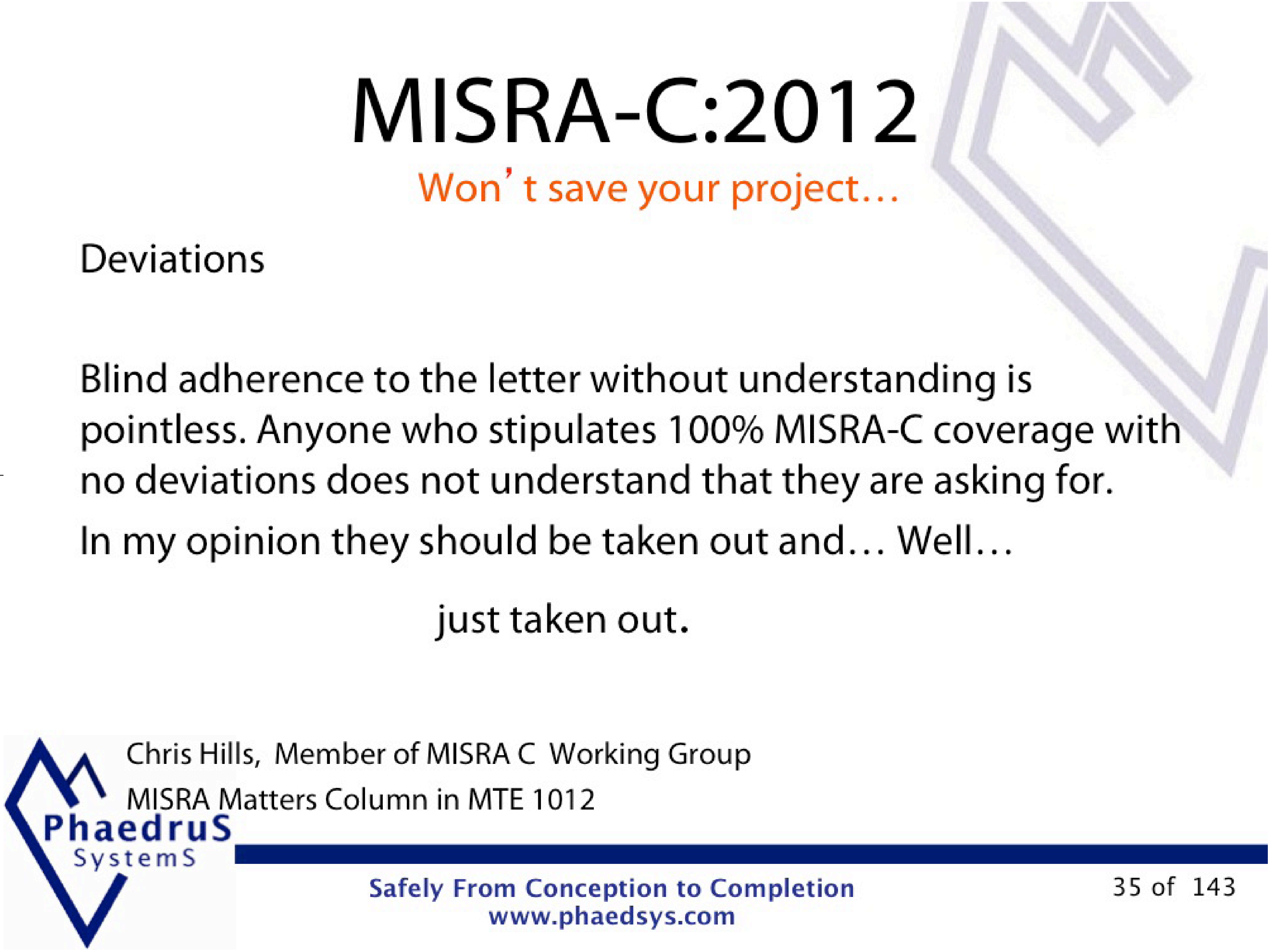
Won't save your project…

MISRA-C DISCLAIMER

Adherence to the requirements of this document does not in itself ensure error-free robust software or guarantee portability and reuse.

Safely From Conception to Completion
www.phaedsys.com

5 of 143



MISRA-C:2012

Won't save your project…

Deviations

Blind adherence to the letter without understanding is pointless. Anyone who stipulates 100% MISRA-C coverage with no deviations does not understand that they are asking for.

In my opinion they should be taken out and… Well…

just taken out.

Chris Hills, Member of MISRA C Working Group

MISRA Matters Column in MTE 1012

PhaedruS SystemS

Safely From Conception to Completion
www.phaedsys.com

35 of 143

MISRA C:2012 Workshop
Device Developer Conference
May 2013

First edition May 2013

Phaedrus Systems Library

The Phaedrus Systems Library is a collection of useful technical documents on development. This includes project management, integrating tools like PC-lint to IDE's, the use of debuggers, coding tricks and tips. The Library also includes the QuEST series.

Copies of this paper (and subsequent versions) with the associated files, will be available with other members of the Library, at:

http://library.phaedsys.com

PhaedruS
SystemS

The Art in Embedded Systems comes through Engineering discipline.