A-B-C: Agile, BMWs, and C++

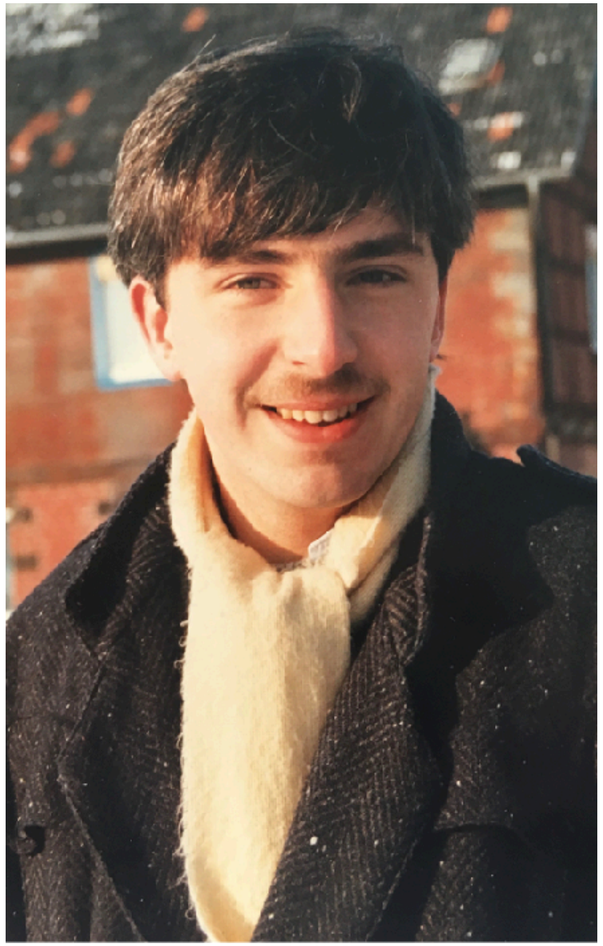
35 years of programming, teaching and still learning

peter.sommerlad@hsr.ch, @PeterSommerlad, www.cevelop.com

More like a full alphabet

- Agile
- Beautiful Code
- C++, Cevelop
- Design Patterns
- Eclipse
- Functional
- Garbage Collection
- History
- IDEs
- Janitor, Java
- Komplexität
- Languages
- Memory
- <numeric>
- Object-Orientation
- Patterns
- Question Authority
- Refactoring
- Simplicity
- Template Visualization
- Unit Testing
- Values
- WikiWikiWeb
- Xtreme Programming
- YAGNI
- Zero Termination

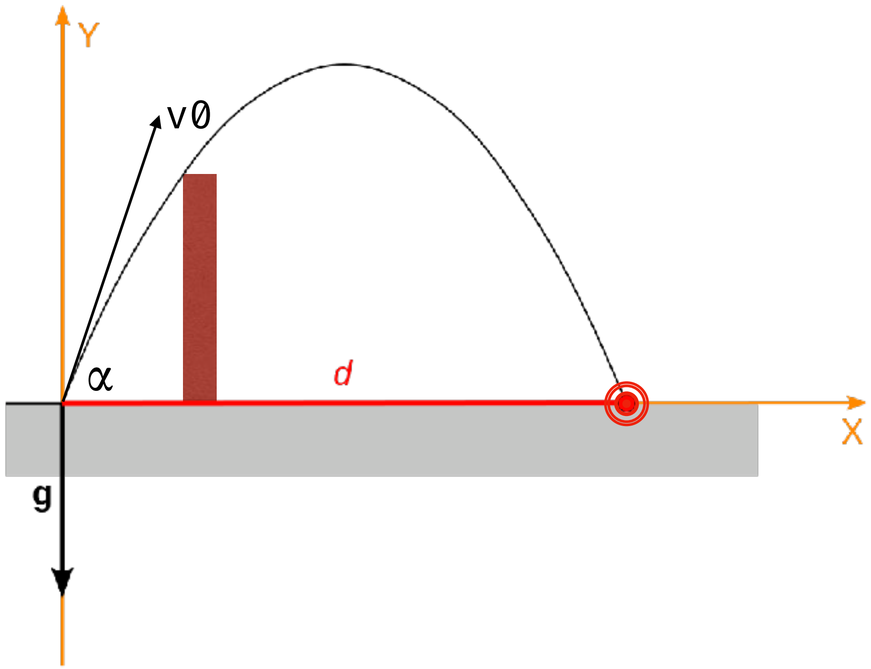
35 years of programming, teaching and still learning



1983: My First Own Program

Line	Instruction	Keycode
01	RCL	34
02	2	02
03	g	32
04	X = Y	-07
05	f	31
06	X < = Y	-35
07	RDN	23
08	STO	33
09	-	51
10	2	02
11	5	05
12	-	51
13	STO	33
14	3	03
15	2	02
16	%	84
17	RCL	34
18	0	00
19	+	61
20	RCL	34
21	1	01
22	+	61
23	STO	33
24	0	00
25	0	00
26	f	31
27	X < = Y	-29
28	GTO 46	-46
29	RCL	34
30	3	03
31	STO	33
32	+	61
33	1	01
34	GTO 48	-48
35	RCL	34
36	1	01
37	g	32
38	X^2	42
39	RCL	34
40	0	00
41	1	01
42	0	00
43	STO	33
44	1	01
45	RDN	23
46	FIX	24
47	9	09
48	RCL	34
49	1	01

Img Source: http://www.hpmuseum.org/3qs/33c3q.jpg

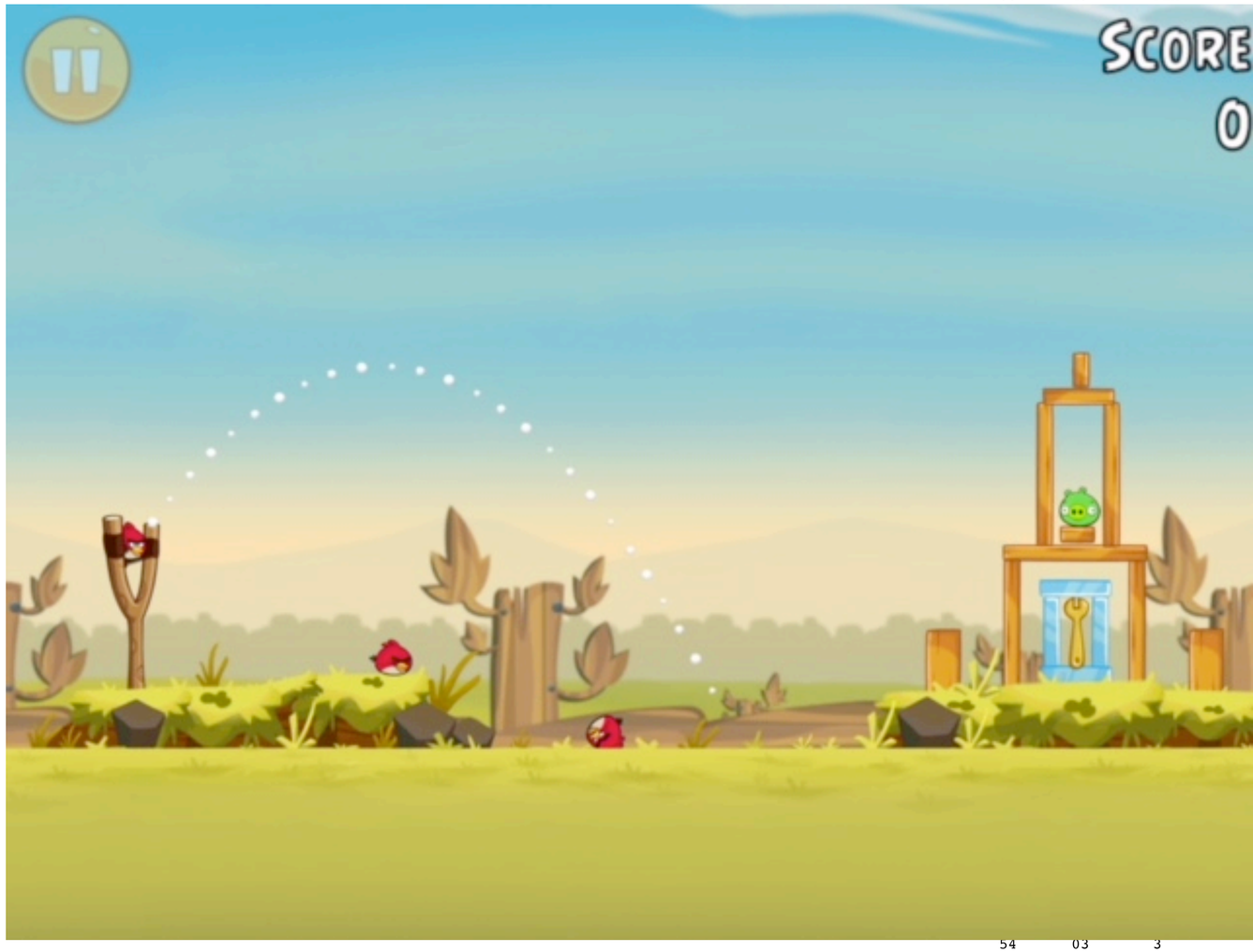


Line	Keycode	Instruction
00	49	FIX
01	00	0
02	56	CP
03	47	X >= T
04	00	0
05	07	7
06	00	0
07	32	X <> T
08	07	7
09	05	5
10	47	X >= T
11	01	1
12	04	4
13	32	X <> T
14	34	RCL
15	00	0
16	74	-
17	32	X <> T
18	12	INV
19	47	X >= T
20	02	2
21	03	3
22	32	X <> T
23	33	STO
24	03	3
25	94	=
26	33	STO
27	00	0
28	34	RCL
29	02	2
30	84	+
31	34	RCL
32	01	1
33	84	+
34	52	(
35	34	RCL
36	03	3
37	74	-
38	05	5
39	53	)
40	35	SUM
41	01	1
42	54	/
43	02	2
44	94	=
45	33	STO
46	02	2
47	56	CP
48	37	X = T
49	07	7
50	08	8
51	93	+/-
52	47	X >= T
53	09	9
54	03	3
55	34	RCL
56	00	0
57	37	X = T
58	08	8
59	00	0
60	34	RCL

1983: My First Own Program

Line	Instruction	Keycode
01	RCL	34
02	2	02
03	g	32
04	X = Y	-07
05	f	31
06	X < = Y	-35
07	RDN	23
08	STO	33
09	-	51
10	2	02
11	5	05
12	-	51
13	STO	33
14	3	03
15	2	02
16	%	28
17	RCL	34
18	0	00
19	+	61
20	RCL	34
21	1	01
22	+	61
23	STO	33
24	0	00
25	0	00
26	f	31
27	X < = Y	-29
28	GTO 46	-46
29	RCL	34
30	3	03
31	STO	33
32	+	61
33	1	01
34	GTO 48	-48
35	RCL	34
36	1	01
37	g	32
38	X^2	42
39	RCL	34
40	0	00
41	1	01
42	0	00
43	STO	33
44	1	01
45	RDN	23
46	FIX	24
47	9	09
48	RCL	34
49	1	01

Img Source: http://www.hpmuseum.org/3qs/33c3q.jpg



Line	Keycode	Instruction
00	49	FIX
54	03	3
55	34	RCL
56	00	0
57	37	X = T
58	08	8
59	00	0
60	34	RCL

Lessons Learned

- Mentally "execute" code
- Using a stack
- Reading foreign code
- Porting code
- Old simple ideas remain useful

1983: My first own Computer

Img Source: https://upload.wikimedia.org/wikipedia/commons/3/33/ZXSpectrum48k.jpg - Bill Bertram 2005 CC SA
Img casette recorder: source: Enrico Grämer

1983: My first own Computer

Img Source: https://upload.wikimedia.org/wikipedia/commons/3/33/ZXSpectrum48k.jpg - Bill Bertram 2005 CC SA
Img casette recorder: source: Enrico Grämer

1983: My first own Computer

Img Source: https://upload.wikimedia.org/wikipedia/commons/3/33/ZXSpectrum48k.jpg - Bill Bertram 2005 CC SA
Img casette recorder: source: Enrico Grämer

1983: My first own Computer

Img Source: https://upload.wikimedia.org/wikipedia/commons/3/33/ZXSpectrum48k.jpg - Bill Bertram 2005 CC SA
Img casette recorder: source: Enrico Grämer

Lessons Learned

- BASIC & Z80 assembly and machine code & Pascal
- Keyboard shortcuts and meta-layouts
- Patience
- Deal with errors of the hardware

1985: First Job - First PC

Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke
and old-computers.com, B+ tree by Grundprinzip

1985: First Job - First PC

Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke and old-computers.com, B+ tree by Grundprinzip

1985: First Job - First PC

Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke and old-computers.com, B+ tree by Grundprinzip

1985: First Job - First PC

Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke and old-computers.com, B+ tree by Grundprinzip

1985: First Job - First PC

Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke and old-computers.com, B+ tree by Grundprinzip

1985: First Job - First PC

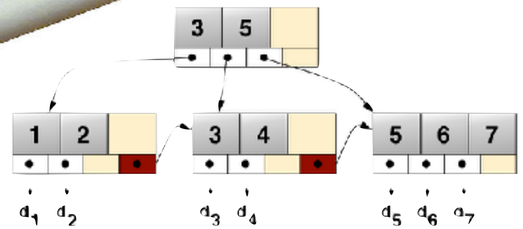


Image source: https://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg Ruben de Rijcke
and old-computers.com, B+ tree by Grundprinzip

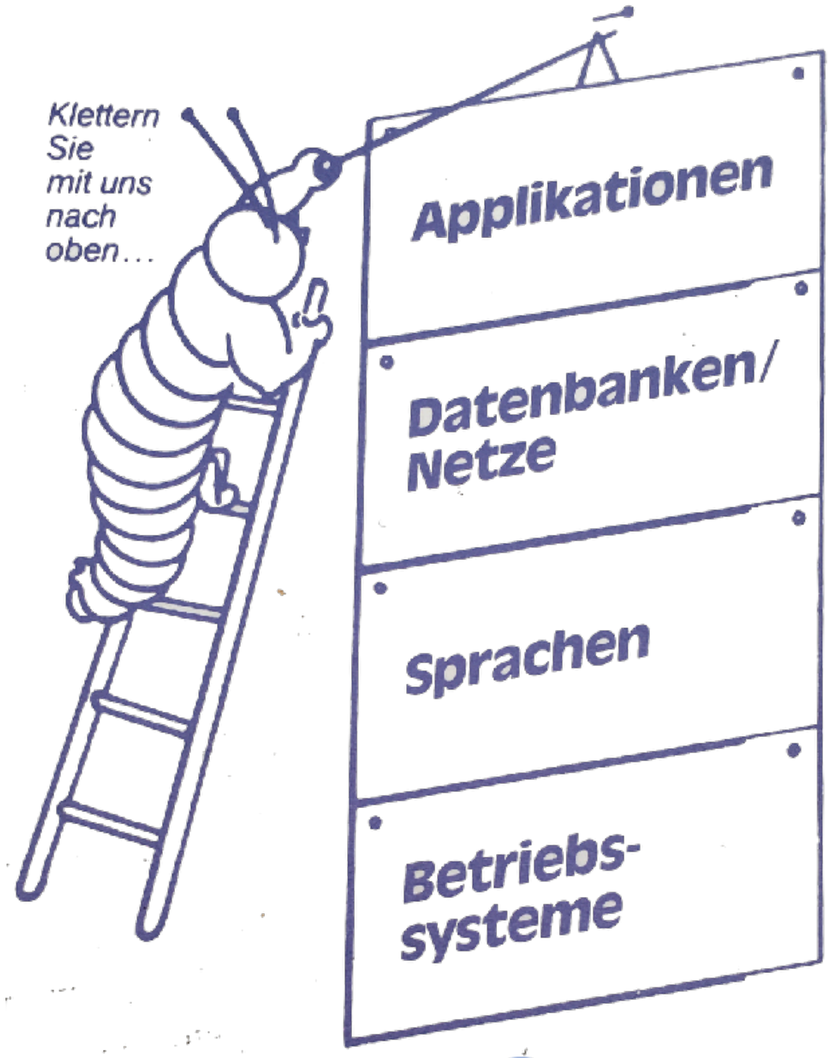
Lessons Learned

- Do not trust programming books and commercial libraries implicitly
- When putting code in books, always compile and test it
- Programming should be supported by a fast and powerful editor
- You should not need to type all of your code
- Generic data-driven solutions make code simpler
- Automate tests

Lessons Learned

- Do not trust programming books and commercial libraries implicitly
- When putting code in books, always compile and test it
- Programming should be supported by a fast and powerful editor
- You should not need to type all of your code
- Generic data-driven solutions make code simpler
- Automate tests

1987-89: C and UNIX courses



Freund+Dirks
empowering your competence

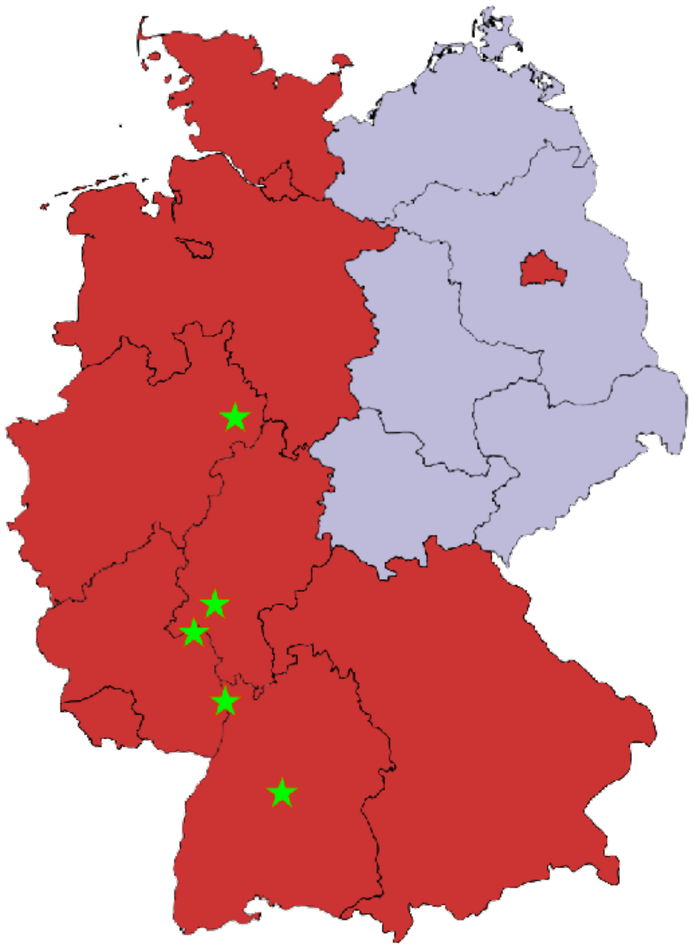
NIXDORF
COMPUTER

IBM®

1988: Some C guidelines

- A function is too long if it does not fit the 80x24 terminal
- A function should not take more than 3-5 parameters, use structs to combine them logically
- Everything is an `int`, unless it is a pointer or a `double`
- Program Abstract Data Types using opaque structs and pointers
- Do not use `scanf()` in production code, check the result of `sscanf()`

1987-89: C and UNIX courses



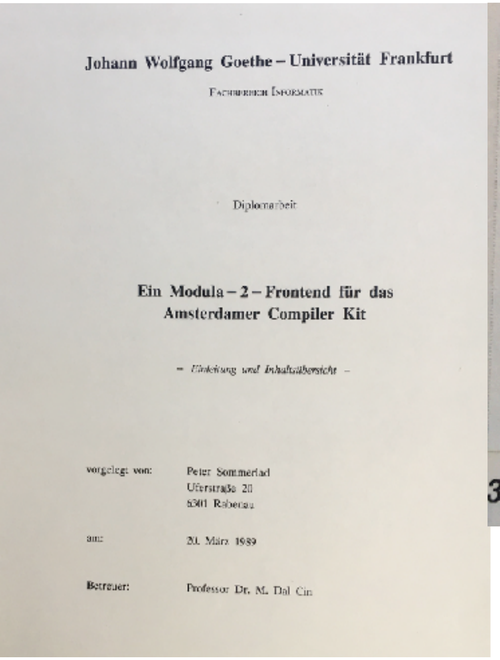
Lessons Learned

Lessons Learned

- C (K+R), UNIX, sh, awk, sed, vi, nroff, make, lex, yacc,...
- You learn the most about a subject by teaching it to others
- One can program in FORTRAN/BASIC/COBOL in any language
- Abstraction, Layering, Cohesion
- "The power of plain text", Regular Expressions, Pipelines, Scripting

1989: Graduation - Diploma Thesis

A Modula-2 Frontend for the Amsterdamer Compiler Kit



Johann Wolfgang Goethe – Universität Frankfurt

Fachbereich Informatik

Diplomarbeit

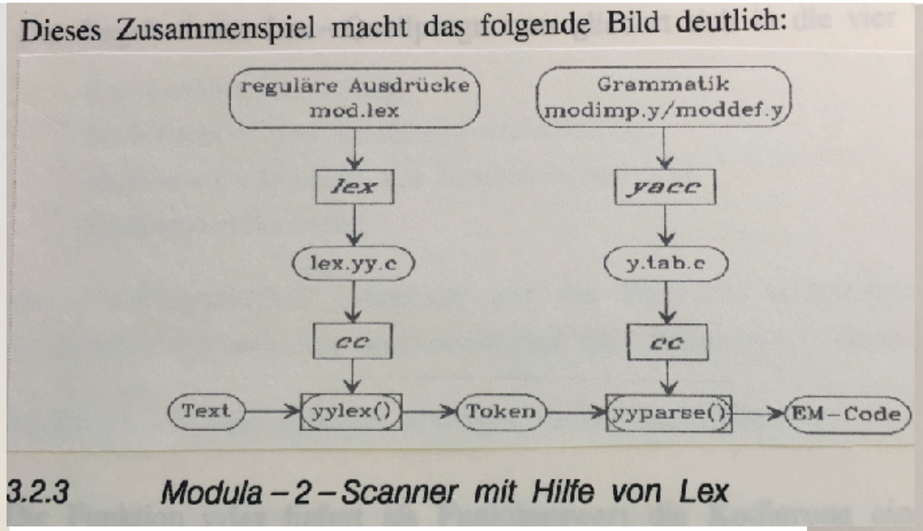
Ein Modula – 2 – Frontend für das Amsterdamer Compiler Kit

– *Einleitung und Inhaltsübersicht* –

vorgelegt von: Peter Sommerlad
Uferstraße 20
6501 Roberau

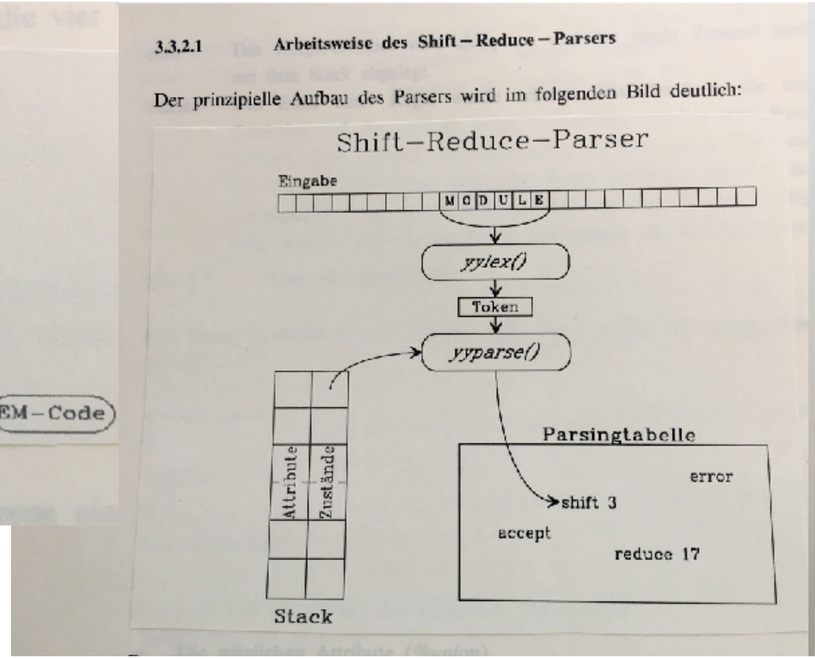
am: 20. März 1989

Betreuer: Professor Dr. M. Dal Cin



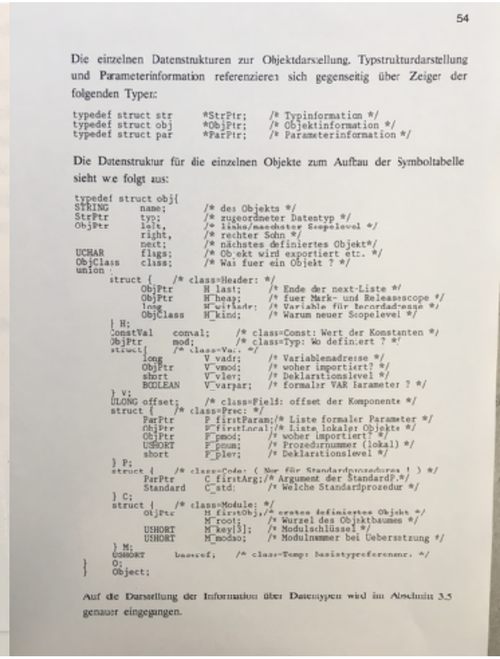
Dieses Zusammenspiel macht das folgende Bild deutlich:

3.2.3 *Modula – 2 – Scanner mit Hilfe von Lex*



3.3.2.1 **Arbeitsweise des Shift – Reduce – Parsers**

Der prinzipielle Aufbau des Parsers wird im folgenden Bild deutlich:



Die einzelnen Datenstrukturen zur Objektdarsellung. Typstrukturdarstellung und Parameterinformation referenzieren sich gegenseitig über Zeiger der folgenden Typen:

```
typedef struct str      *StrPtr;    /* Typinformation */
typedef struct obj      *ObjPtr;    /* Objektinformation */
typedef struct par      *ParPtr;    /* Parameterinformation */
```

Die Datenstruktur für die einzelnen Objekte zum Auftau der Symboltabelle sieht we folgt aus:

```
typedef struct obj{
STRING      name;       /* des Objekts */
StrPtr      typ;        /* zugeordneter Datentyp */
ObjPtr      left,       /* links/naechster Scopelevel */
            right,      /* rechter Sohn */
            next;       /* naechstes definiertes Objekt*/
UCHAR       flags;      /* Objekt wird exportiert etc. */
ObjClass    class;      /* Was fuer ein Objekt ? */
union {
        struct {    /* class=Header: */
            ObjPtr      M_last;     /* Ende der next-Liste */
            ObjPtr      M_heap;     /* fuer Mark- und Releasescope */
            long        M_varadr;   /* Variable für Recordadresse */
            ObjClass    M_kind;     /* Warum neuer Scopelevel */
        } H;
ConstVal    conval;     /* class=Const: Wert der Konstanten */
ObjPtr      mod;        /* class=Typ: Wo definiert ? */
struct{     /* class=Var: */
            long        V_vadr;     /* Variablenadresse */
            ObjPtr      V_vmod;     /* woher importiert? */
            short       V_vlev;     /* Deklarationslevel */
            BOOLEAN     V_varpar;   /* formalsr VAR Parameter ? */
} V;
ULONG offset;   /* class=Field: offset der Komponente */
struct {    /* class=Proc: */
            ParPtr      P_firstParam;/* Liste formaler Parameter */
            ObjPtr      P_firstLocal;/* Liste lokaler Objekte */
            OBJPtr      P_pmod;     /* woher importiert? */
            USHORT      P_pnum;     /* Prozedurnummer (lokal) */
            short       P_plev;     /* Deklarationslevel */
} P;
struct {    /* class=Code: ( Nur für Standardprozeduren ! ) */
            ParPtr      C_firstArg;/* Argument der StandardP.*/
            Standard    C_std;      /* Welche Standardprozedur */
} C;
struct {    /* class=Module: */
            objPtr      M_firstObj;/* erstes definiertes Objekt */
            M_root;     /* Wurzel des Objektbaumes */
            USHORT      M_key[3];   /* Modulschlüssel */
            USHORT      M_modno;    /* Modulnummer bei Uebersetzung */
} M;
UBHORT      basetef,    /* class=Temp: Basistypreferenznr. */
} O;
Object;
```

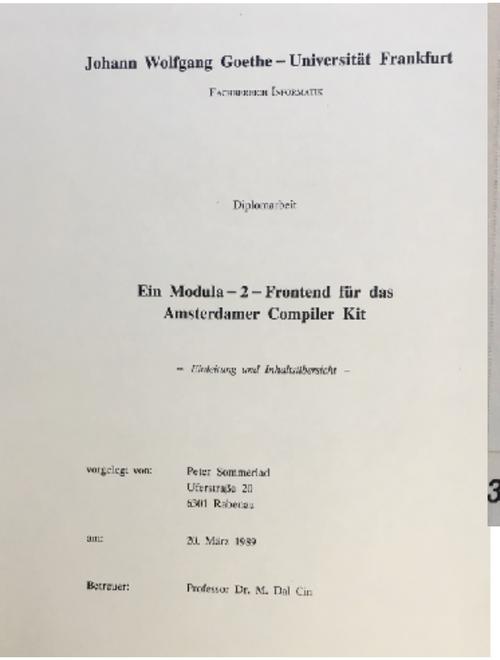
Auf die Darstellung der Information über Datentypen wird im Abschnitt 3.5 genauer eingegangen.

- also first C++ experiments with Zortech C++ in 1989

1989: Graduation -

A Modula-2 Fron

Amsterdamer C



Johann Wolfgang Goethe – Universität Frankfurt

Fachbereich Informatik

Diplomarbeit

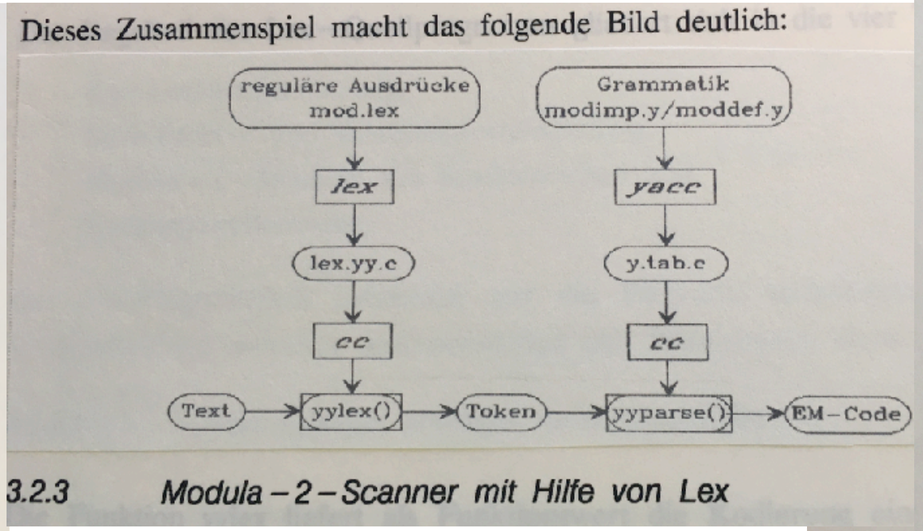
Ein Modula – 2 – Frontend für das Amsterdamer Compiler Kit

– Einleitung und Inhaltsübersicht –

vorgelegt von: Peter Sommerlad
Uferstraße 20
6501 Roberau

am: 20. März 1989

Betreuer: Professor Dr. M. Dal Cin



- also first C++ experiments with Zortech

Die einzelnen Datenstrukturen zur Objektdarstellung, Typstrukturdarstellung und Parameterinformation referenzieren sich gegenseitig über Zeiger der folgenden Typen:

```
typedef struct str      *StrPtr;    /* Typinformation */
typedef struct obj      *ObjPtr;    /* Objektinformation */
typedef struct par      *ParPtr;    /* Parameterinformation */
```

Die Datenstruktur für die einzelnen Objekte zum Aufbau der Symboltabelle sieht wie folgt aus:

```
typedef struct obj{
STRING      name;       /* des Objekts */
StrPtr      typ;        /* zugeordneter Datentyp */
ObjPtr      left,       /* links/naechster Scopelevel */
            right,      /* rechter Sohn */
            next;       /* nächstes definiertes Objekt*/
UCHAR       flags;      /* Objekt wird exportiert etc. */
ObjClass    class;      /* Was fuer ein Objekt ? */
union {
      struct {    /* class=Header: */
            ObjPtr      H_last;     /* Ende der next-Liste */
            ObjPtr      H_heap;     /* fuer Mark- und Releasescope */
            long        H_withadr;  /* Variable für Recordadresse */
            ObjClass    H_kind;     /* Warum neuer Scopelevel */
      } H;
      ConstVal    conval;     /* class=Const: Wert der Konstanten */
      ObjPtr      mod;        /* class=Typ: Wo definiert ? */
      struct{     /* class=Var: */
            long        V_vadr;     /* Variablenadresse */
            ObjPtr      V_vmod;     /* woher importiert? */
            short       V_vlev;     /* Deklarationslevel */
            BOOLEAN     V_varpar;   /* formaler VAR Parameter ? */
      } V;
      ULONG offset;     /* class=Field: offset der Komponente */
      struct {    /* class=Proc: */
            ParPtr      P_firstParam;/* Liste formaler Parameter */
            ObjPtr      P_firstLocal;/* Liste lokaler Objekte */
            ObjPtr      P_pmod;     /* woher importiert? */
            USHORT      P_pnum;     /* Prozedurnummer (lokal) */
            short       P_plev;     /* Deklarationslevel */
      } P;
      struct {    /* class=Code: ( Nur für Standardprozeduren ! ) */
            ParPtr      C_firstArg;/* Argument der StandardP.*/
            Standard    C_std;      /* Welche Standardprozedur */
      } C;
      struct {    /* class=Module: */
            ObjPtr      M_firstObj,/* erstes definiertes Objekt */
                        M_root;     /* Wurzel des Objektbaumes */
            USHORT      M_key[3];   /* Modulschlüssel */
            USHORT      M_modno;    /* Modulnummer bei Uebersetzung */
      } M;
      USHORT      baseref;    /* class=Temp: Basistypreferenznr. */
}     O;
}     Object;
```

Auf die Darstellung der Information über Datentypen wird im Abschnitt 3.5 genauer eingegangen.

Lessons Learned

- Compilers and ASTs with polymorphic nodes (union)
- Virtual-Machine languages (P-Code, EM-Code)
- RAM and processing power makes compiles much faster
- ADTs work, even within a compiler
- Lex and Yacc can be used in practice

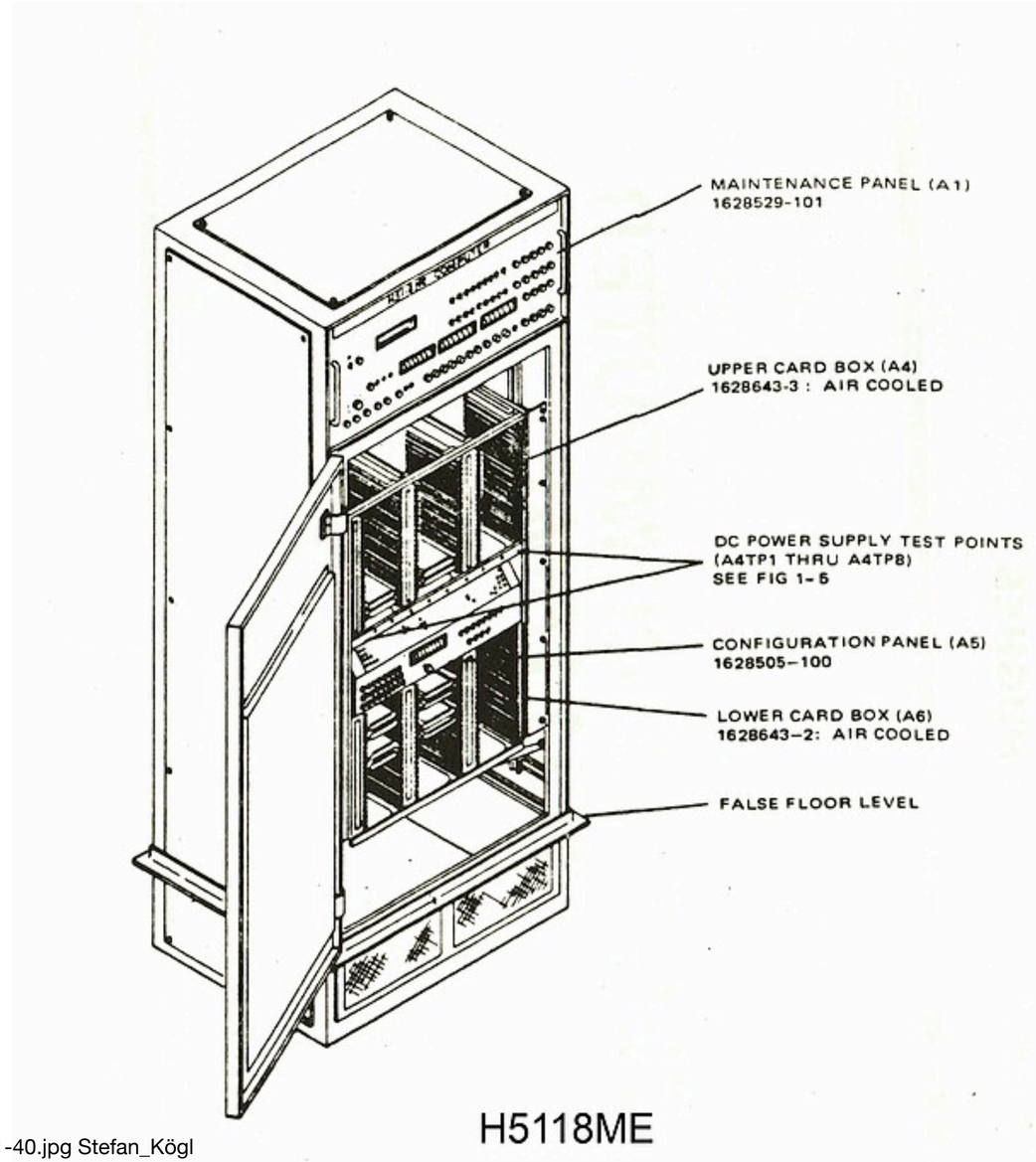
Conscript at Bundeswehr

https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-11-40.jpg Stefan_Kögl
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_drive.jpg Autopilot
http://www.columbia.edu/cu/computinghistory/rp04.html

Conscript at Bundeswehr

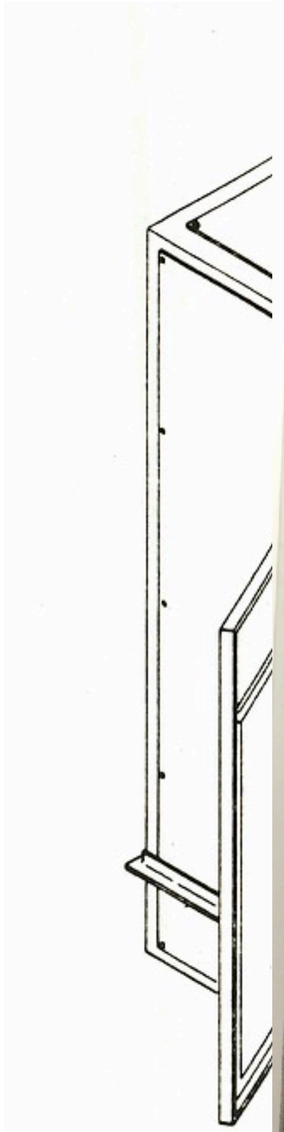
https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-11-40.jpg Stefan_Kögl
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_drive.jpg Autopilot
http://www.columbia.edu/cu/computinghistory/rp04.html

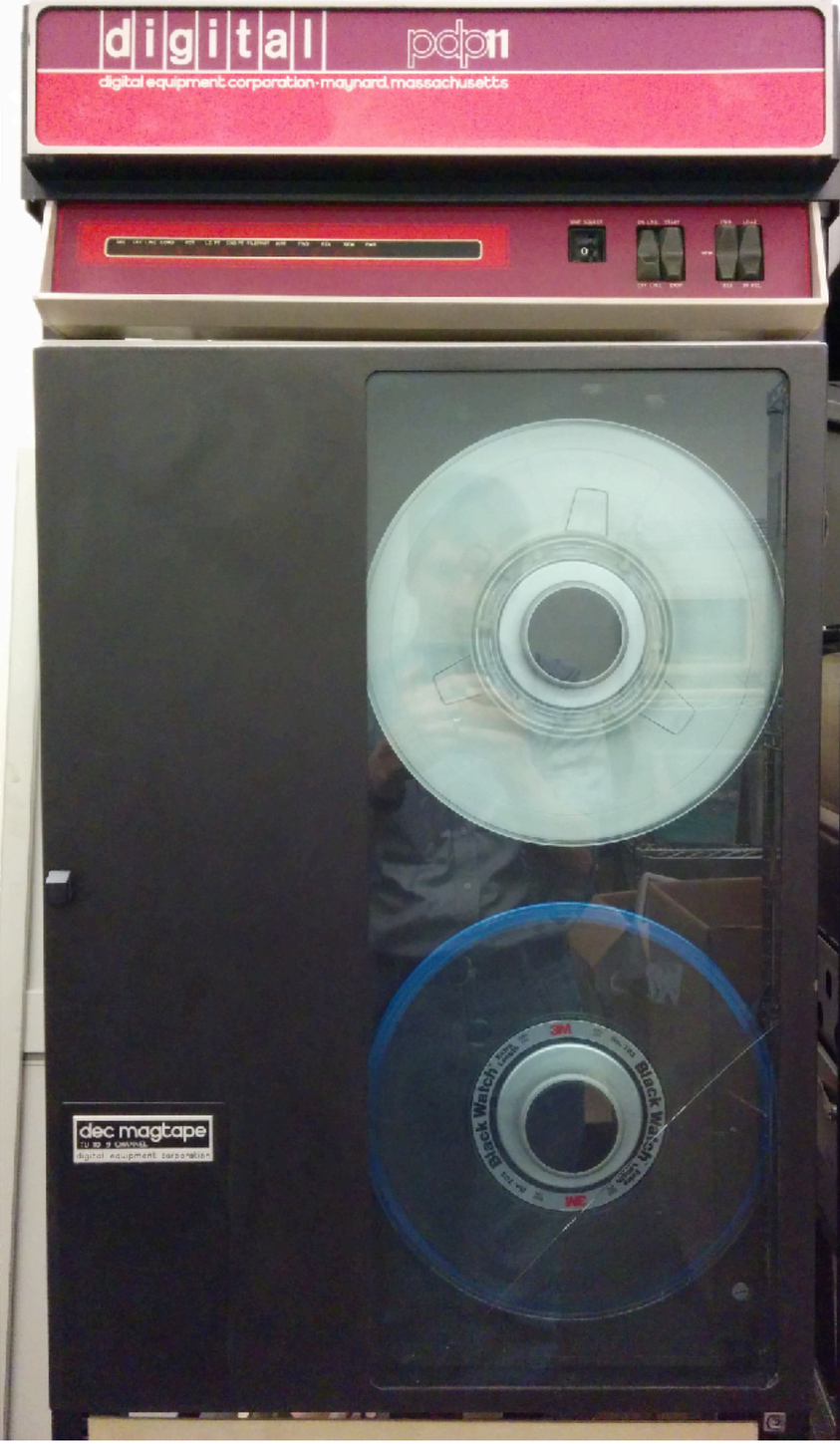
Conscript at Bundeswehr



https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-11-40.jpg Stefan_Kögl
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_drive.jpg Autopilot
http://www.columbia.edu/cu/computinghistory/rp04.html

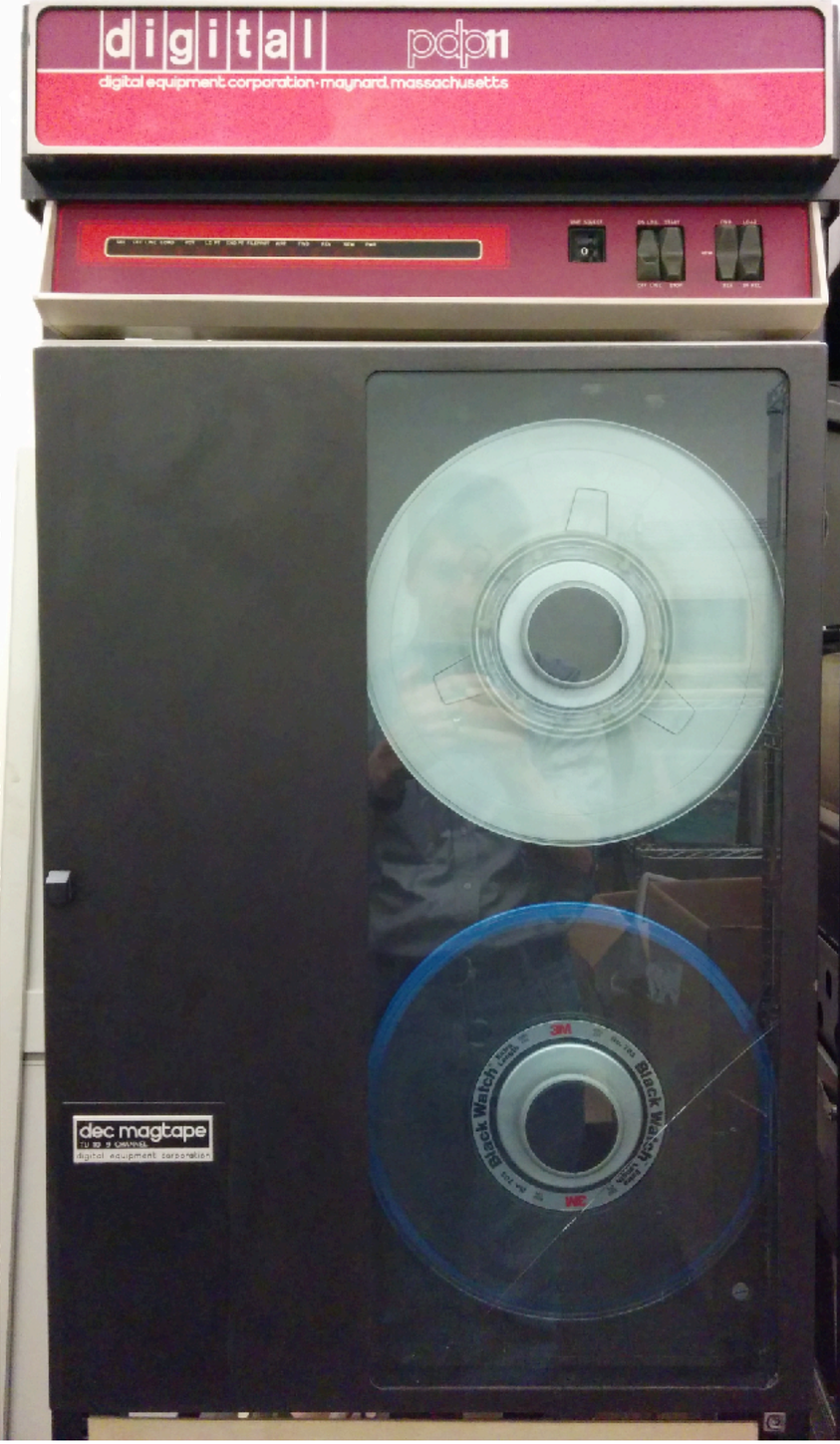
Conscript at Bundesweh





https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-11-40.jpg Stefan_Kögl
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_drive.jpg Autopilot
http://www.columbia.edu/cu/computinghistory/rp04.html

Conscript at Bundesweh



https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-11-40.jpg Stefan_Kögl
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_drive.jpg Autopilot
http://www.columbia.edu/cu/computinghistory/rp04.html

Conscript at Bundes

https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_d
http://www.columbia.edu/cu/computinghistory/rp04.html

LEVEL II

AD A101061

RADC-TR-81-143
Final Technical Report
June 1981

COMPUTER PROGRAMMING MANUAL FOR THE JOVIAL (J73) LANGUAGE

Softech, Inc.

DTIC FILE COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
JUL 6 1981
S D
D

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

Conscript at Bundes

https://upload.wikimedia.org/wikipedia/commons/e/ee/Pdp-
https://commons.wikimedia.org/wiki/File:DEC_TU10_tape_d
http://www.columbia.edu/cu/computinghistory/rp04.html

LEVEL II

AD A101061

RADC-TR-81-143
Final Technical Report
June 1981

COMPUTER PROGRAMMING MANUAL FOR THE JOVIAL (J73) LANGUAGE

Softech, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC FILE COPY

DTIC
ELECTE
JUL 6 1981
S D
D

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

Conscript at Bundes

https://upload.wikimedia.org/wikipe /comm
https://commons.wikimedia.org/wiki/ DEC_TU
http://www.columbia.edu/cu/computing ory/rp04.html

LEVEL II

AD A101061

RADC-TR-81-143
Final Technical Report
June 1981

COMPUTER PROGRAMMING MANUAL FOR THE JOVIAL (J73) LANGUAGE

Softtech, Inc.

DTIC COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
JUL 6 1981
S D
D

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

Conscript at Bundes

https://upload.wikimedia.org/wikipe /comm
https://commons.wikimedia.org/wiki/ DEC_TU
http://www.columbia.edu/cu/computing ory/rp04.html

LEVEL II

AD A101061

RADC-TR-81-143
Final Technical Report
June 1981

COMPUTER PROGRAMMING MANUAL FOR THE JOVIAL (J73) LANGUAGE

Softtech, Inc.

DTIC COPY

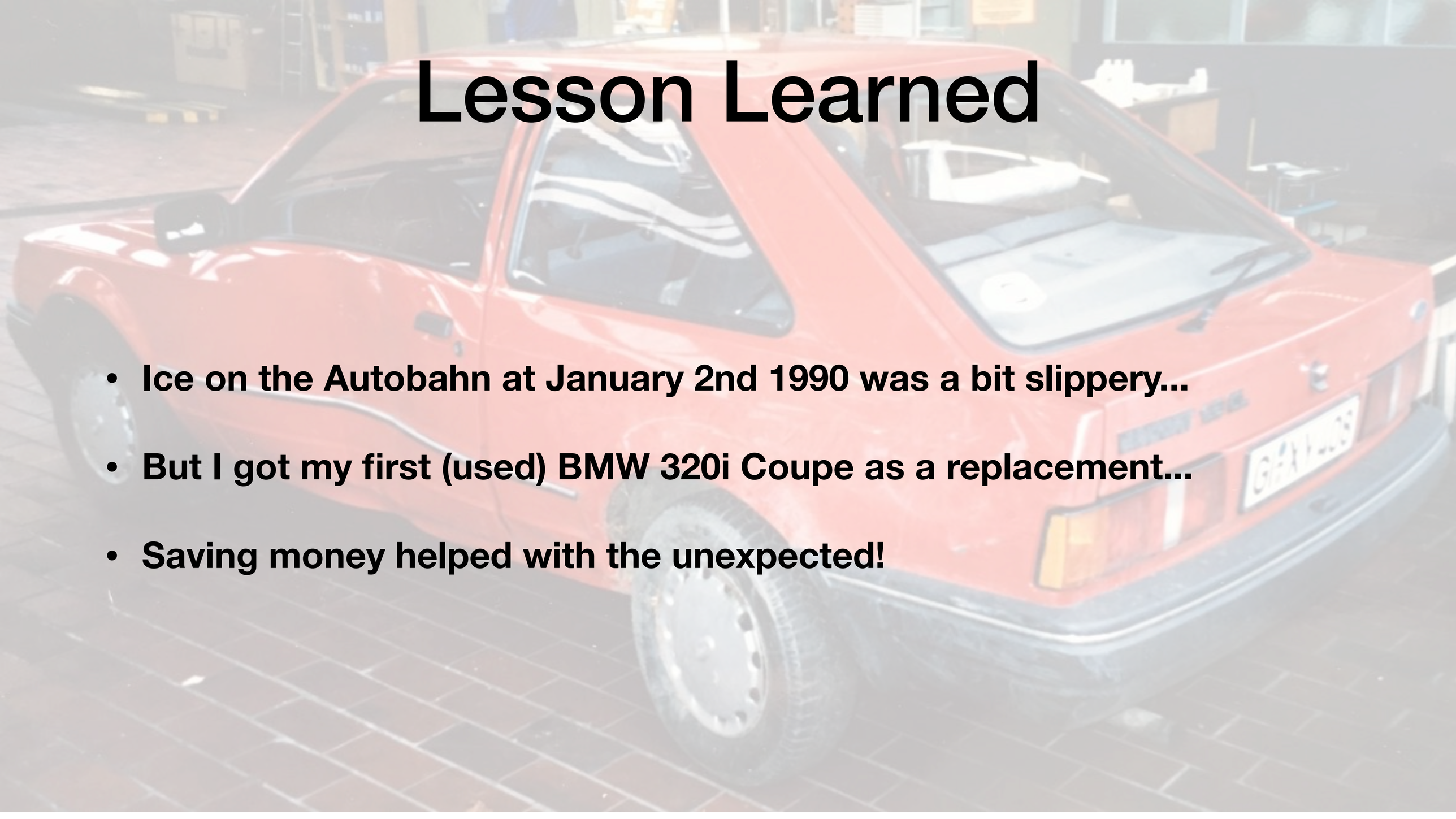
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
JUL 6 1981
S D
D

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

Lesson Learned

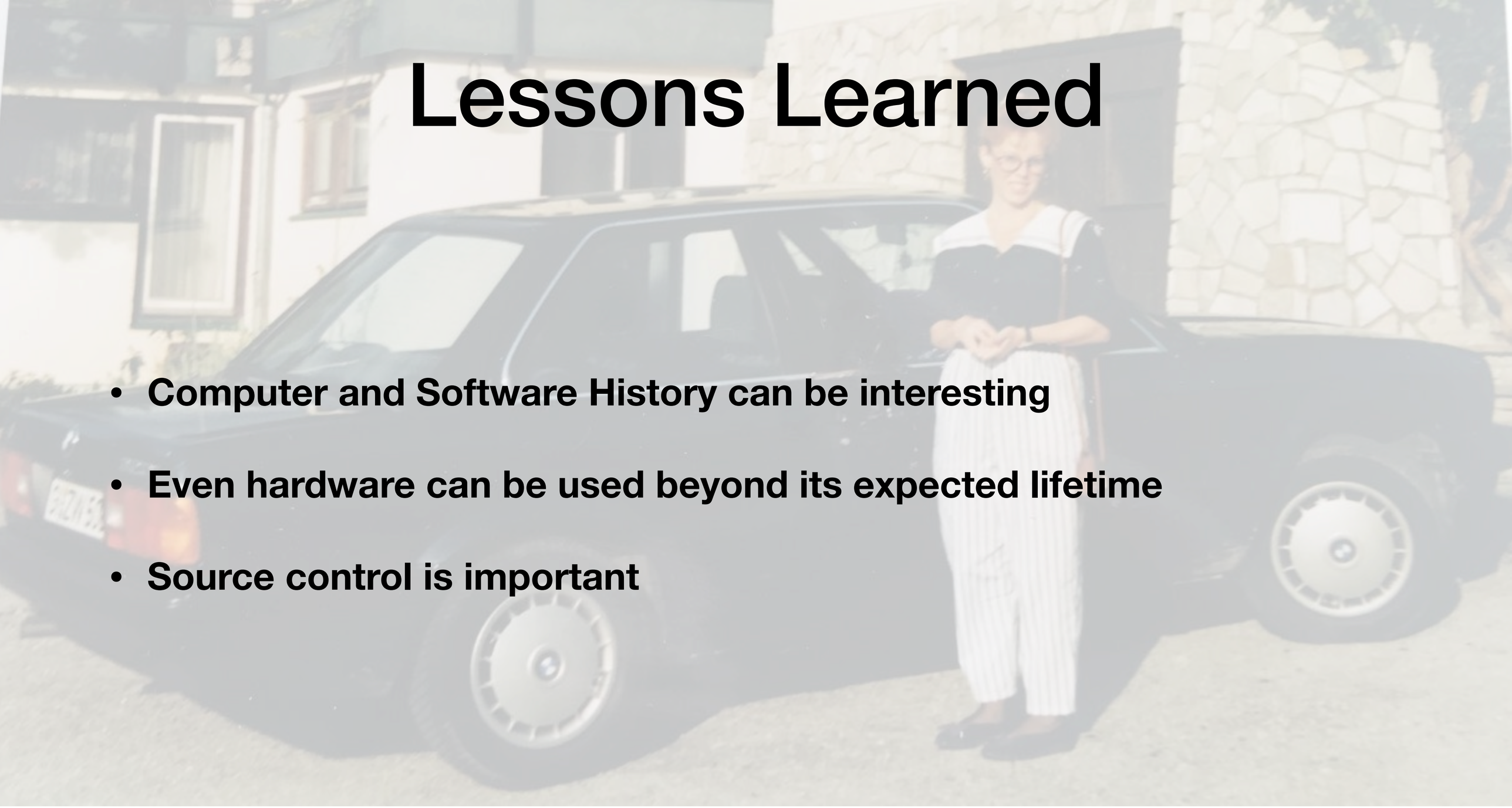
Lesson Learned



- **Ice on the Autobahn at January 2nd 1990 was a bit slippery...**
- **But I got my first (used) BMW 320i Coupe as a replacement...**
- **Saving money helped with the unexpected!**

Lessons Learned

Lessons Learned



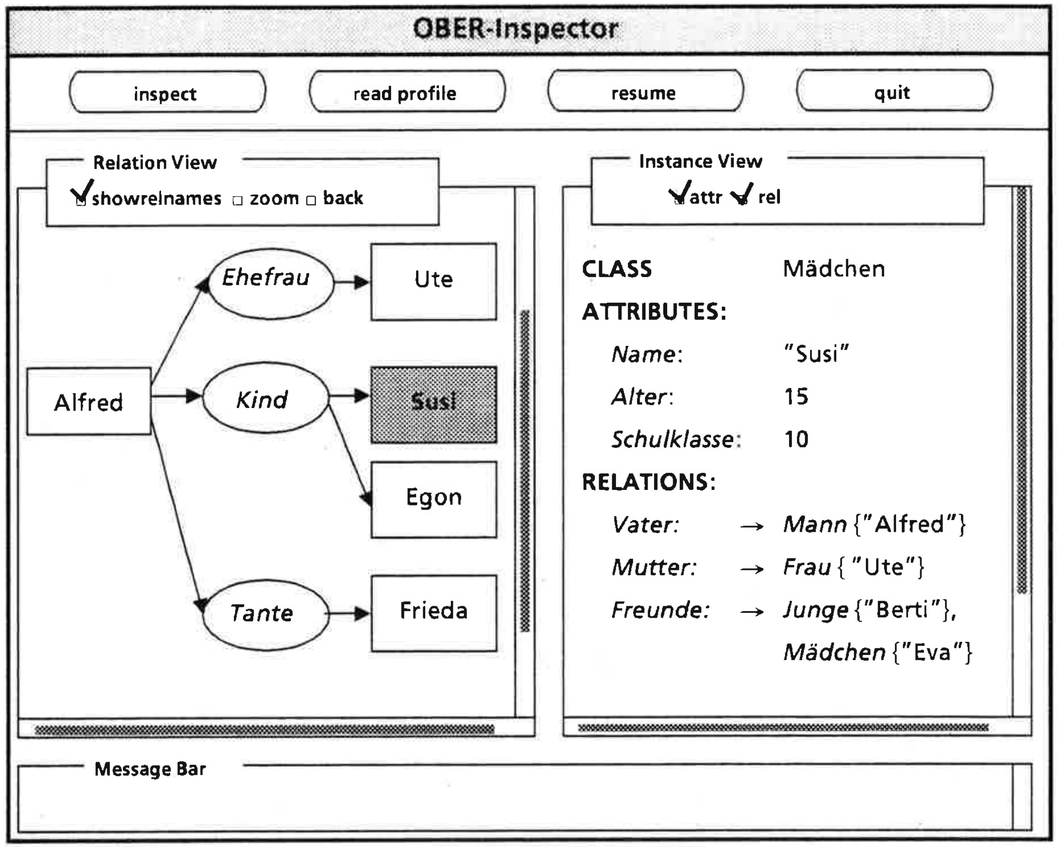
- **Computer and Software History can be interesting**
- **Even hardware can be used beyond its expected lifetime**
- **Source control is important**


SIEMENS

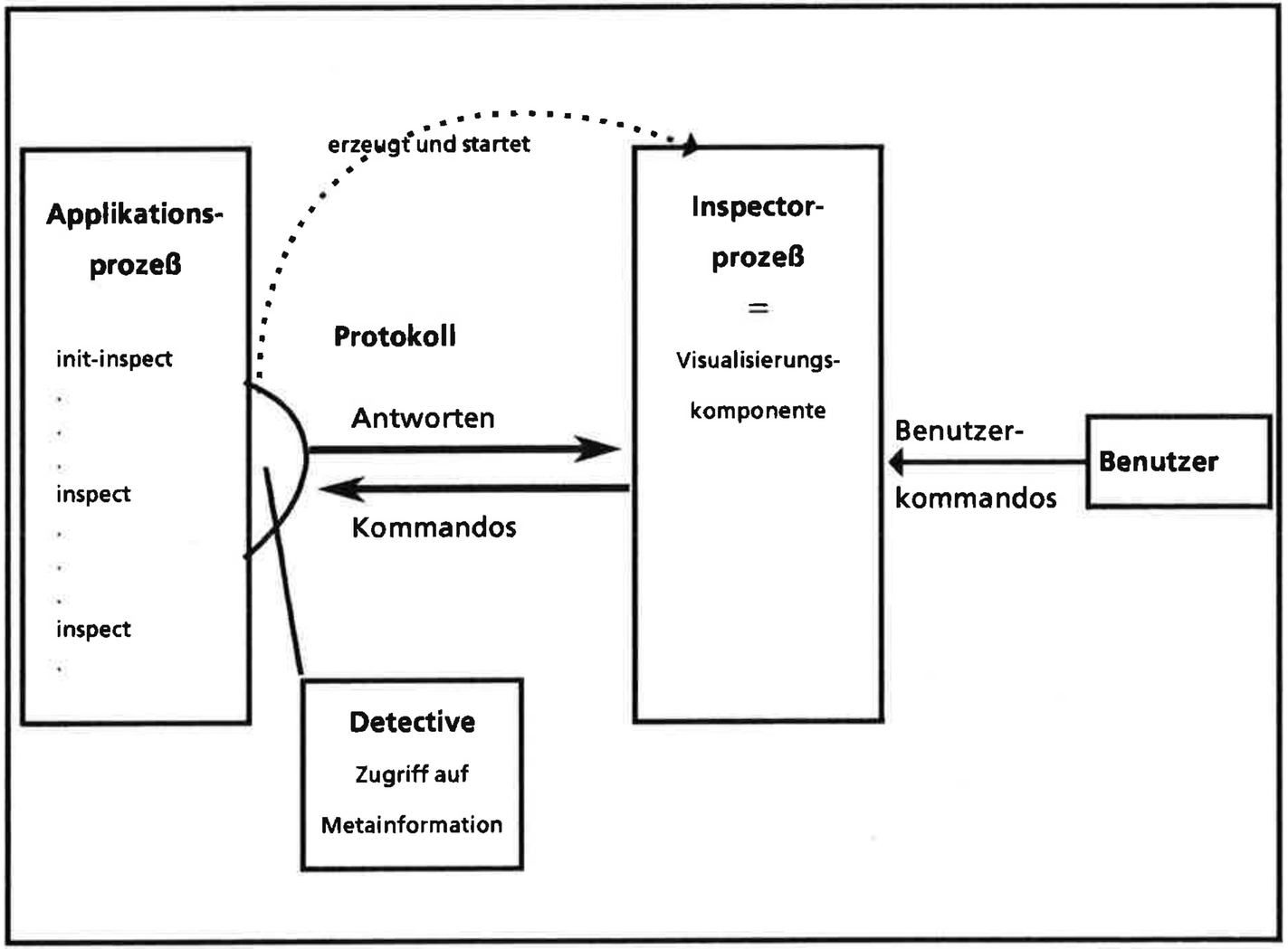
1990: First "real" Job at SIEMENS

- Project I applied for: Object-oriented Programming Environment (OOPU)
  - for C++ in C++, prototype with Erich Gamma/André Weinand's ET++
- Sidetrack: Object-Inspector for TOROS_OBER
  - a C-macro-based object-oriented system
  - TOROS_OBER was used for BMW's diagnostics computers
- OOPU was killed due to internal politics

My first Architecture



- Ober-Inspector
  (nickname: Derrick)



1991: C++ Guidelines

- Some rules still apply
- Some are obsolete
- Some I learned how to do better today

1991: Wedding

1991: Wedding

1993-: Patterns and PLoPs

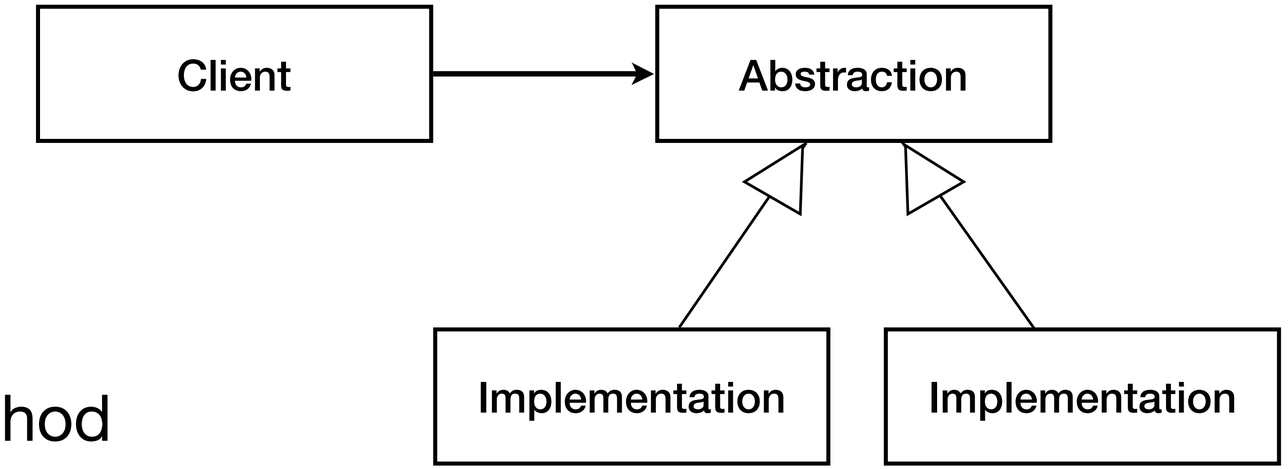
1993-: Patterns and PLoPs

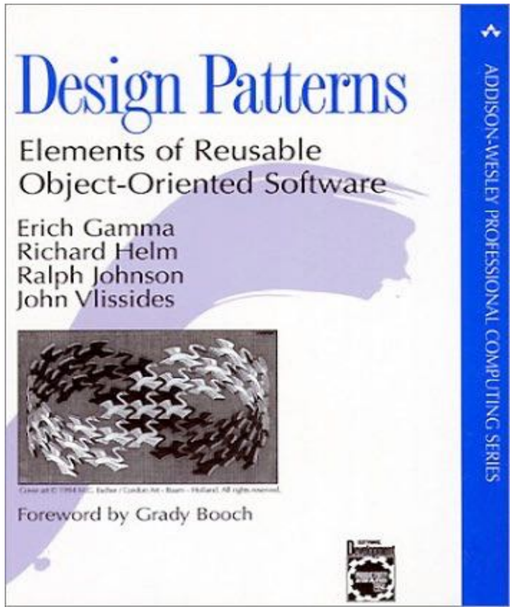
1993-: Patterns and PLoPs

1993-: Patterns and PLoPs

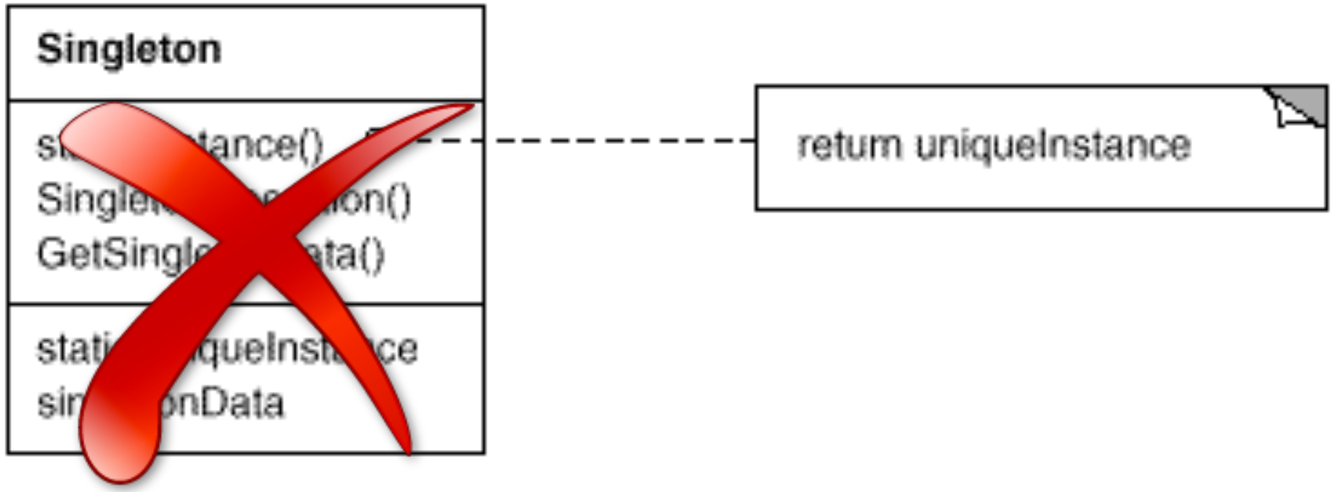
Most Important Design Patterns

- Command
- Strategy
- Template Method with Factory Method
- Composite
- Decorator
- Null Object






AGED 12 YEARS
OF DUFFTOWN
TRADE MARK
THE SINGLETON
Single Malt Scotch Whisky
DUFFTOWN
ESTD 1896
MATURED FOR 12 YEARS
12
PERFECTLY BALANCED, NATURALLY RICH & SMOOTH


Singleton
return uniqueInstance

On Singleton Design Pattern

- **DO NOT USE IT! Think about design!**
- If you need globals, use a const global
- Why came it into being?
- static initialization order is undefined in C++ with non-zero non-trivial initialization


```
class Singleton {
public:
  static Singleton* Instance();
protected:
  Singleton();
private:
  static Singleton* _instance;
};
Singleton* Singleton::_instance = 0;

Singleton* Singleton::Instance() {
  if (_instance == 0) {
    _instance = new Singleton;
  }
  return _instance;
}
```

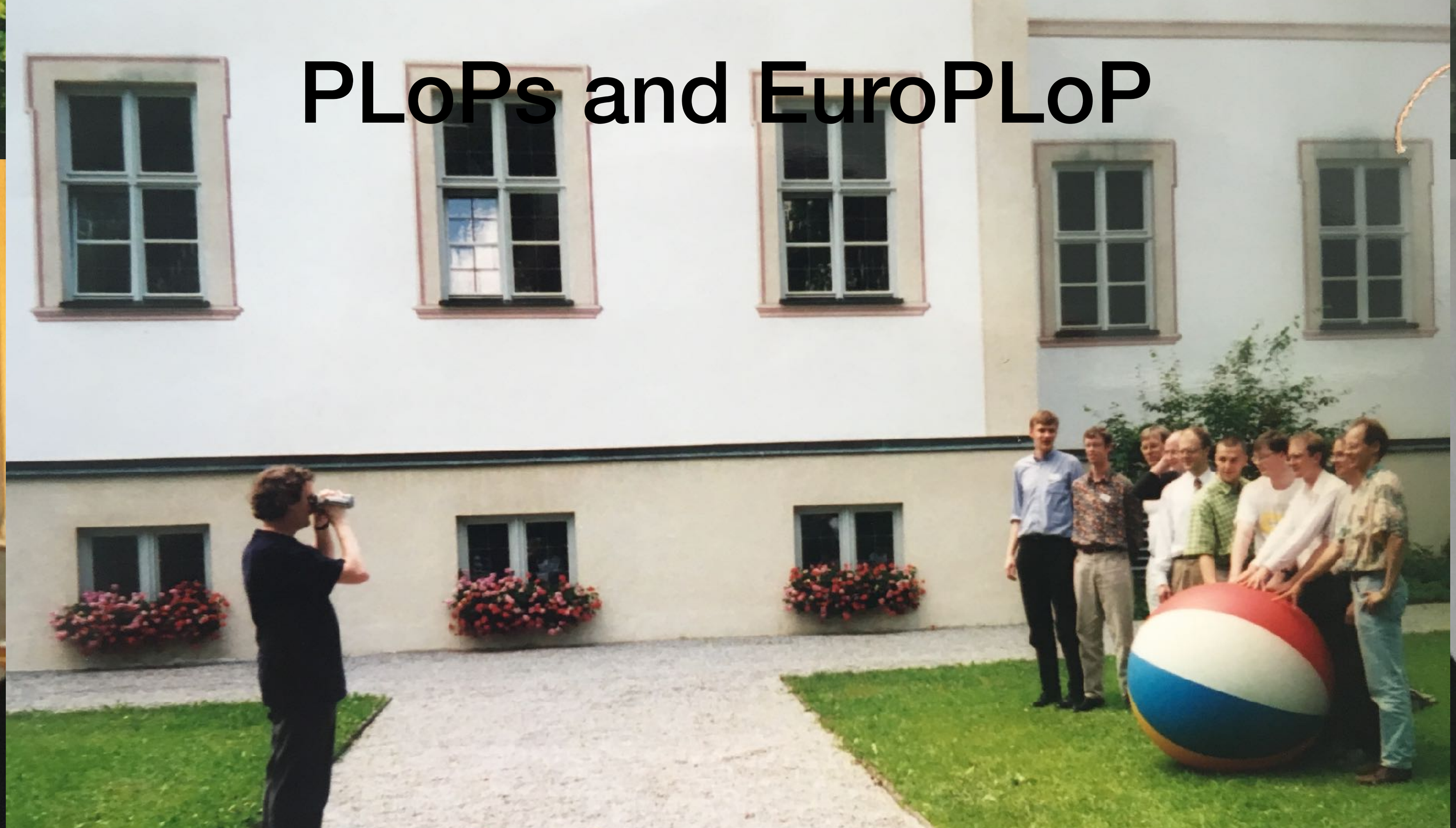
PLoPs and EuroPLoP

PLoPs and EuroPLoP

PLoPs and EuroPLoP

PLoPs and EuroPLoP

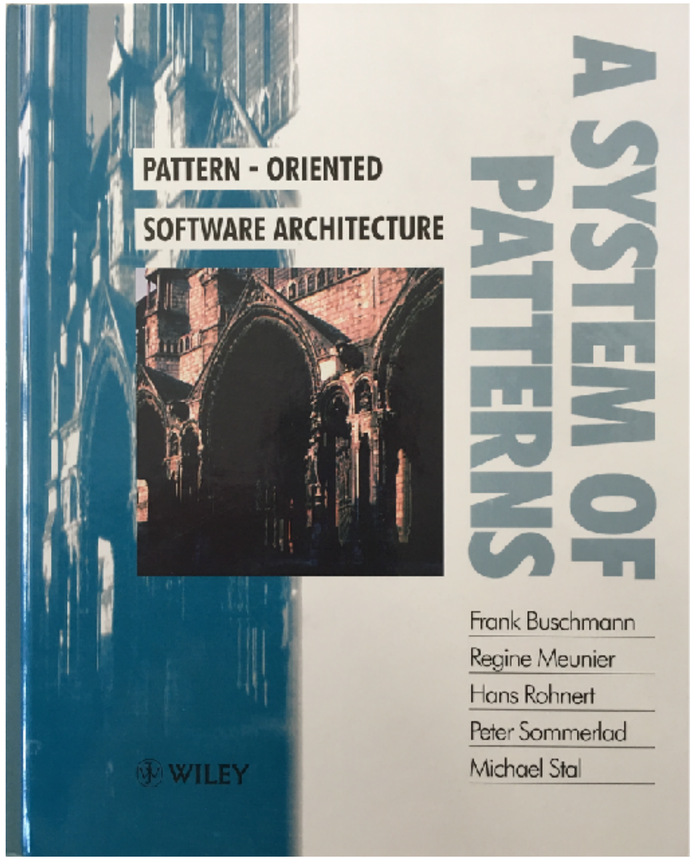
PLoPs and EuroPLoP

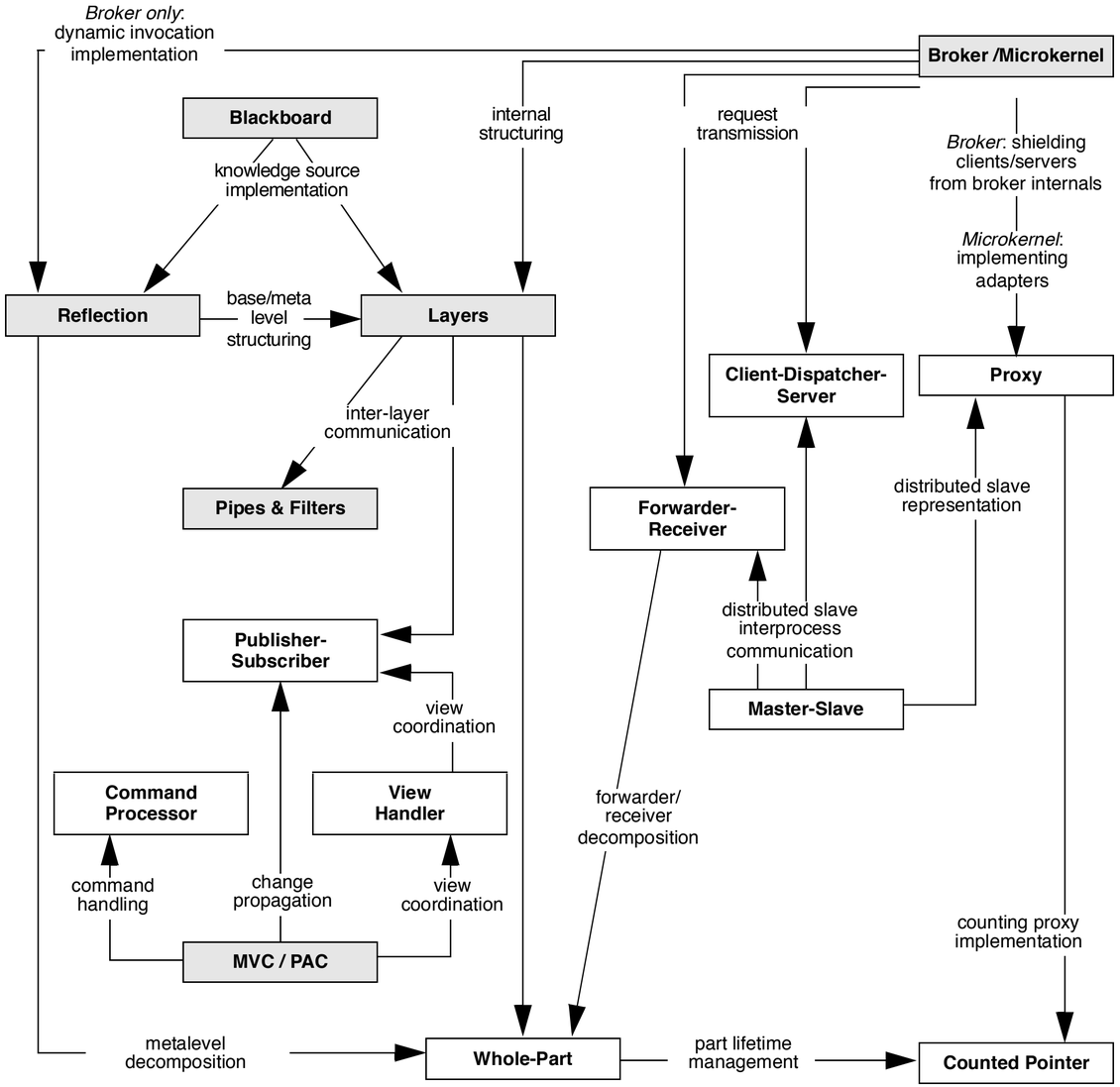


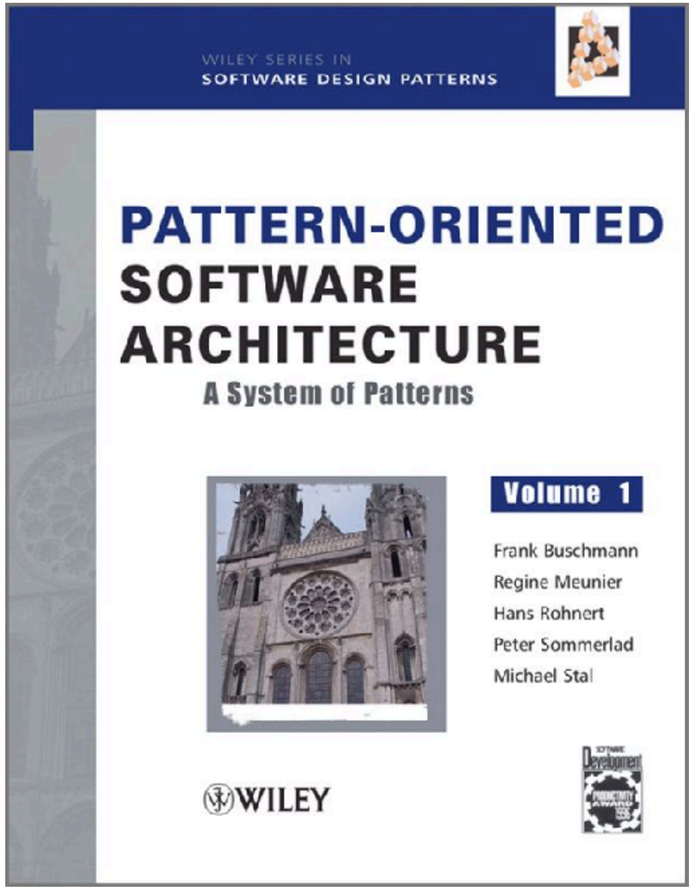

OLD TOWN PIZZ
STANFORD UNIVERSITY


Rebecca
Kyle

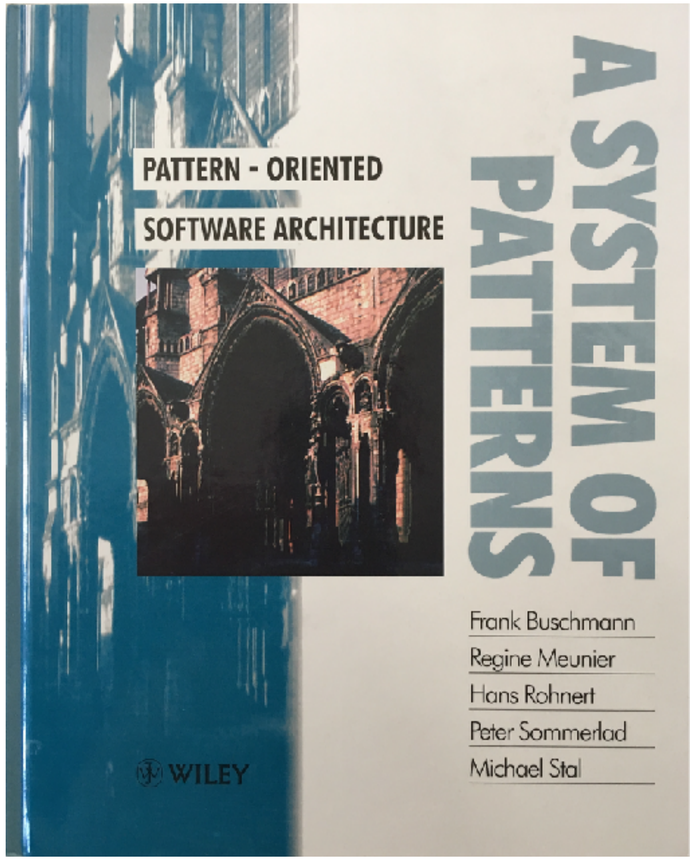
POSA 1

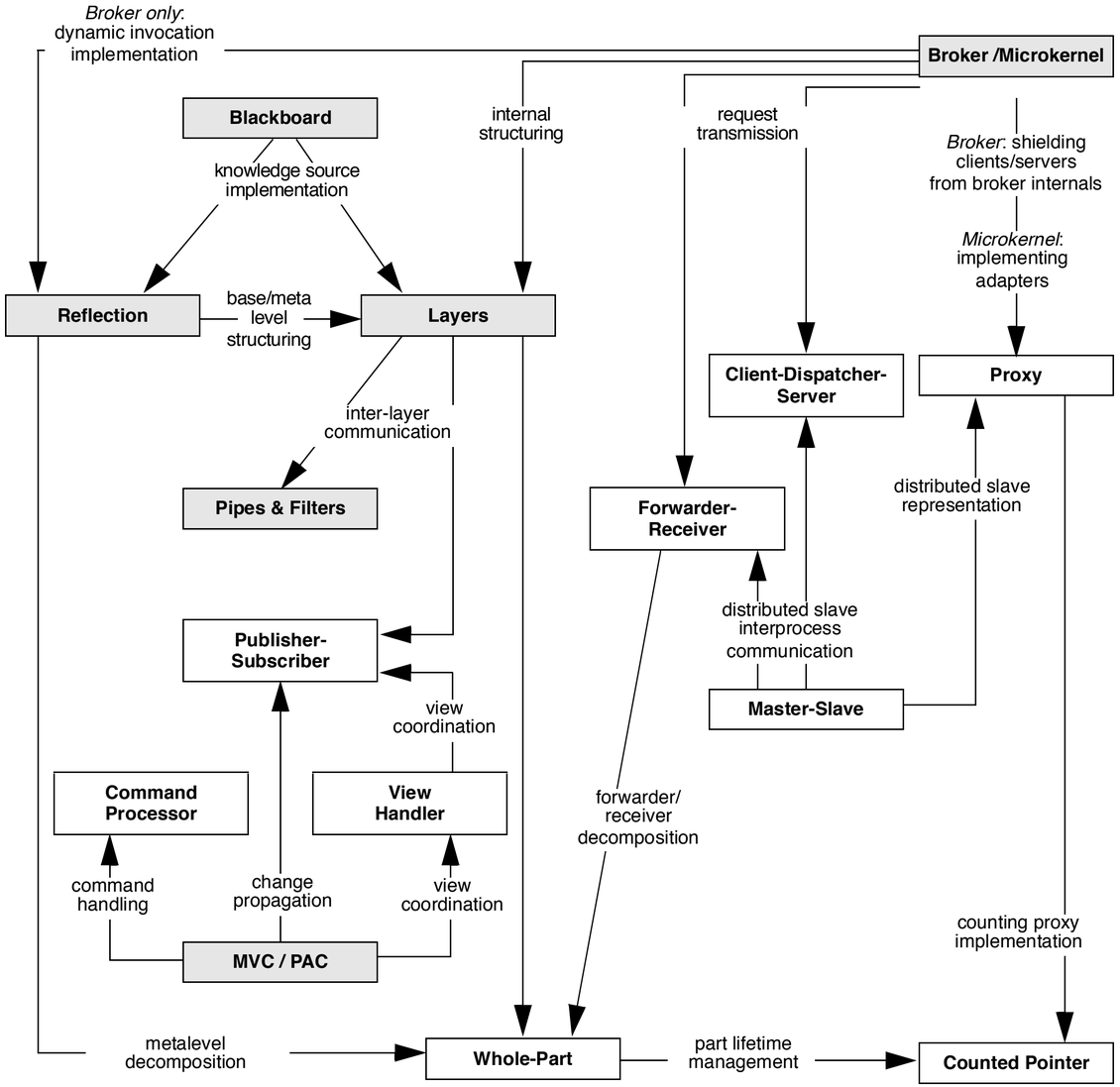

PATTERN - ORIENTED
SOFTWARE ARCHITECTURE
A SYSTEM OF PATTERNS
Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal
WILEY

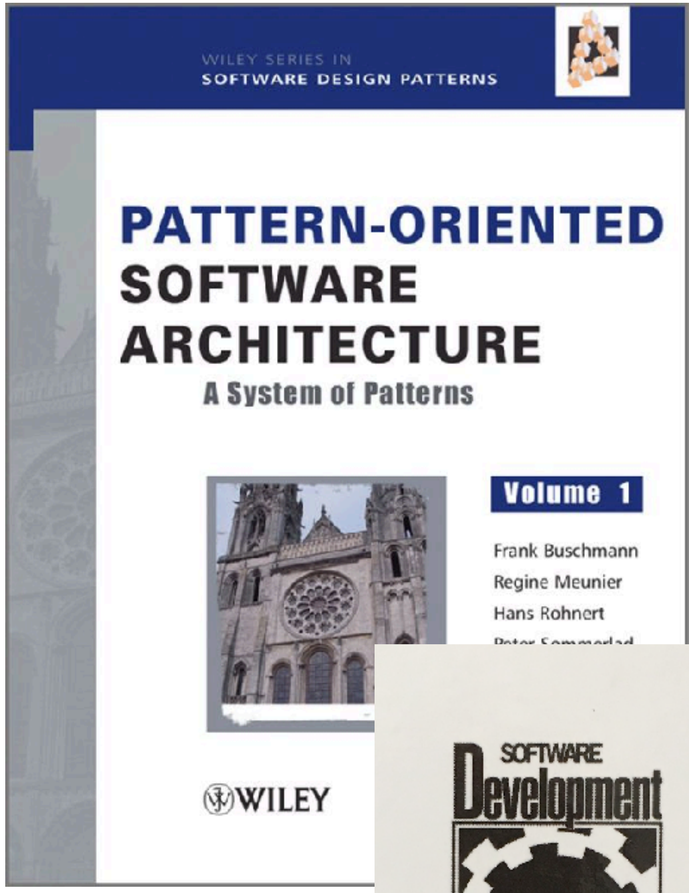

Broker only: dynamic invocation implementation
Broker /Microkernel
Blackboard
knowledge source implementation
internal structuring
request transmission
Broker: shielding clients/servers from broker internals
Microkernel: implementing adapters
Reflection
base/meta level structuring
Layers
Client-Dispatcher-Server
Proxy
inter-layer communication
Pipes & Filters
Forwarder-Receiver
distributed slave representation
distributed slave interprocess communication
Publisher-Subscriber
Master-Slave
view coordination
Command Processor
View Handler
forwarder/ receiver decomposition
command handling
change propagation
view coordination
MVC / PAC
counting proxy implementation
metalevel decomposition
Whole-Part
part lifetime management
Counted Pointer

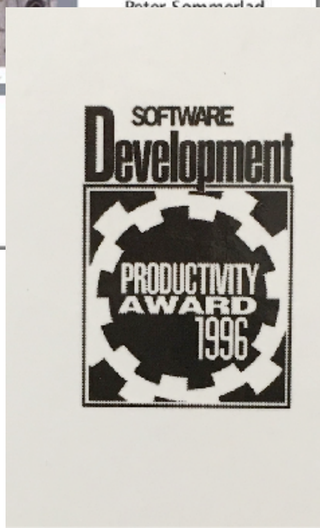

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A System of Patterns
Volume 1
Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal
WILEY

POSA 1


PATTERN - ORIENTED
SOFTWARE ARCHITECTURE
A SYSTEM OF PATTERNS
Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal
WILEY

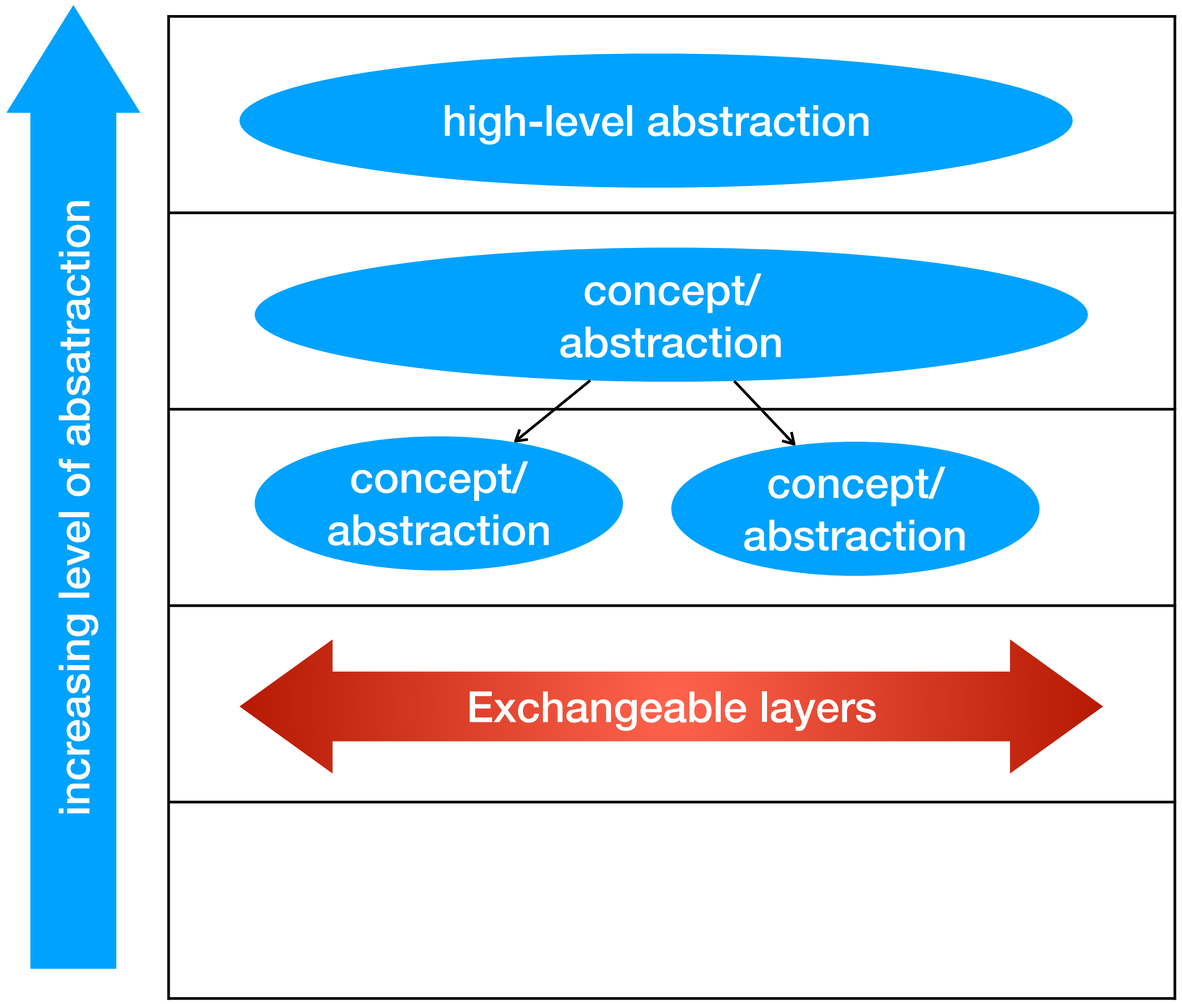

Broker only: dynamic invocation implementation
Broker /Microkernel
Blackboard
knowledge source implementation
internal structuring
request transmission
Broker: shielding clients/servers from broker internals
Microkernel: implementing adapters
Reflection
base/meta level structuring
Layers
Client-Dispatcher-Server
Proxy
inter-layer communication
Pipes & Filters
Forwarder-Receiver
distributed slave representation
distributed slave interprocess communication
Publisher-Subscriber
Master-Slave
view coordination
Command Processor
View Handler
forwarder/ receiver decomposition
command handling
change propagation
view coordination
MVC / PAC
counting proxy implementation
metalevel decomposition
Whole-Part
part lifetime management
Counted Pointer

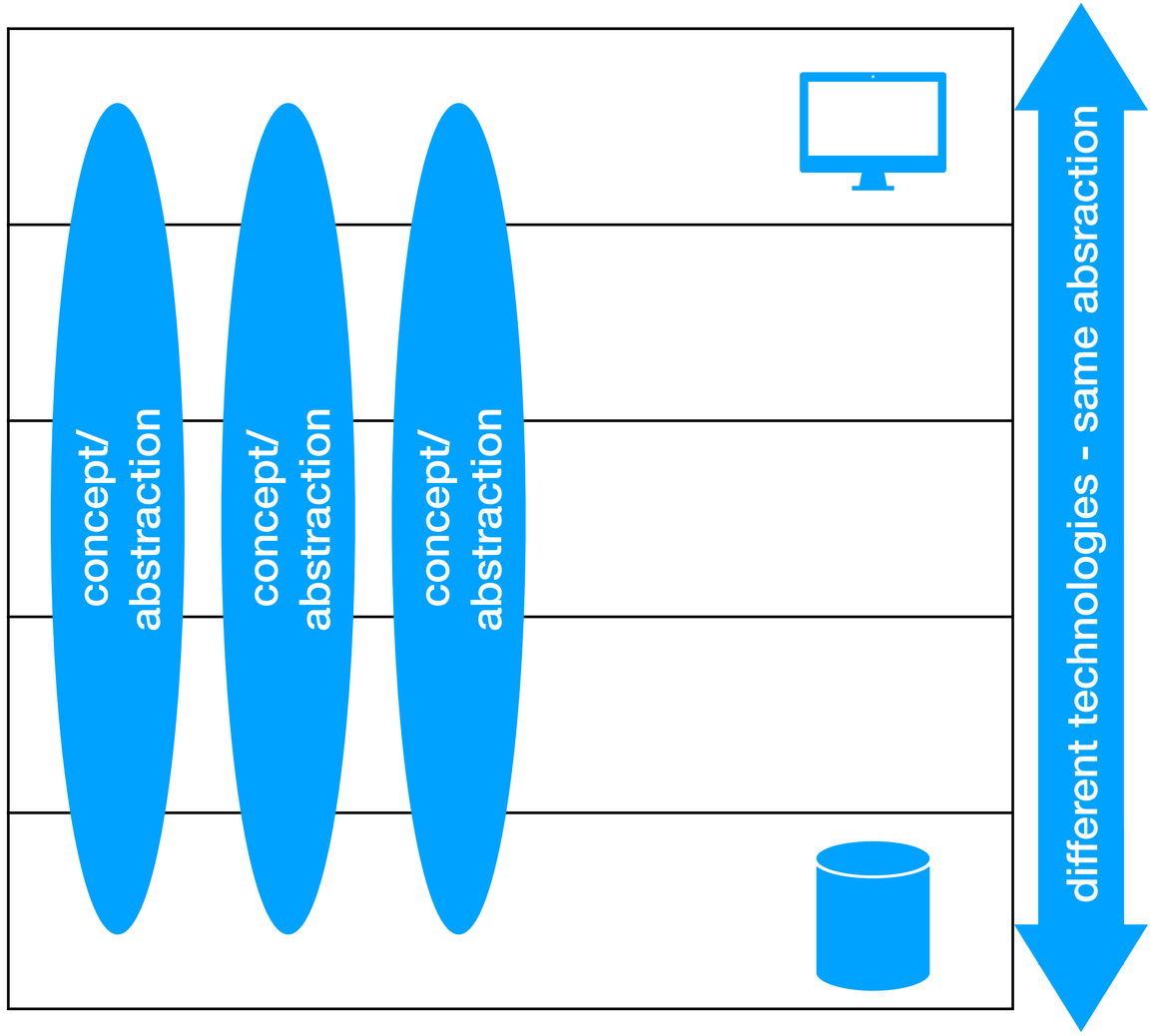

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A System of Patterns
Volume 1
Frank Buschmann
Regine Meunier
Hans Rohnert
WILEY


SOFTWARE
Development
PRODUCTIVITY
AWARD
1996

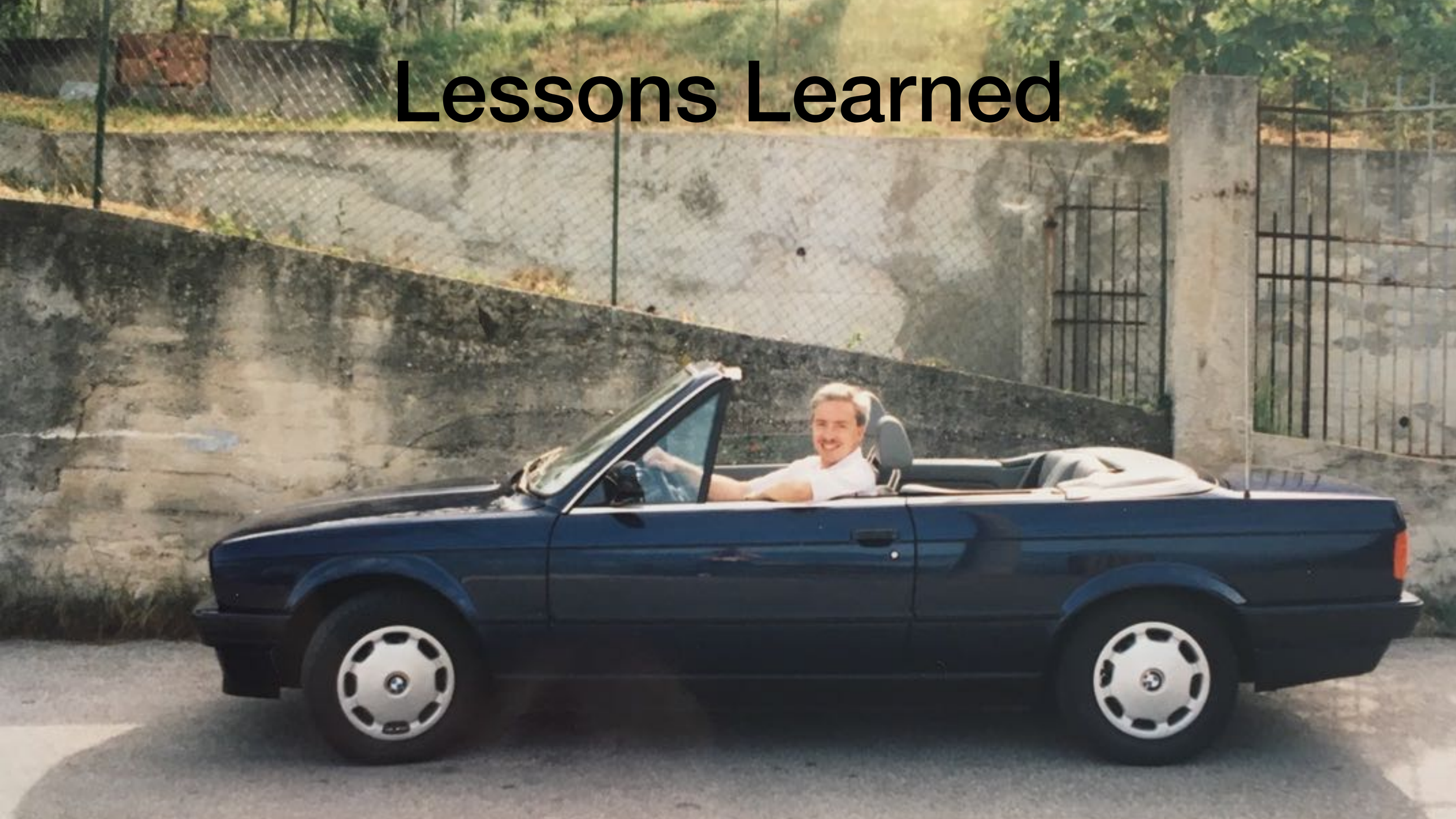
area of change

Layers vs. Tiers





Lessons Learned



Lessons Learned

- "A prophet has no honor in his own country" - get out to be heard
- Good Patterns require several rewrites
- Shepherding teaches you to become a better author
- Get a good copy editor, when publishing
- Network to learn - attend and speak at conferences
- Most important part of conferences are the breaks

1997: Move to Switzerland

1997: Move to Switzerland

- Hollywood Principle applied:
  I called Erich Gamma:
  "can I work with you?"
  he said "I'll call you back"

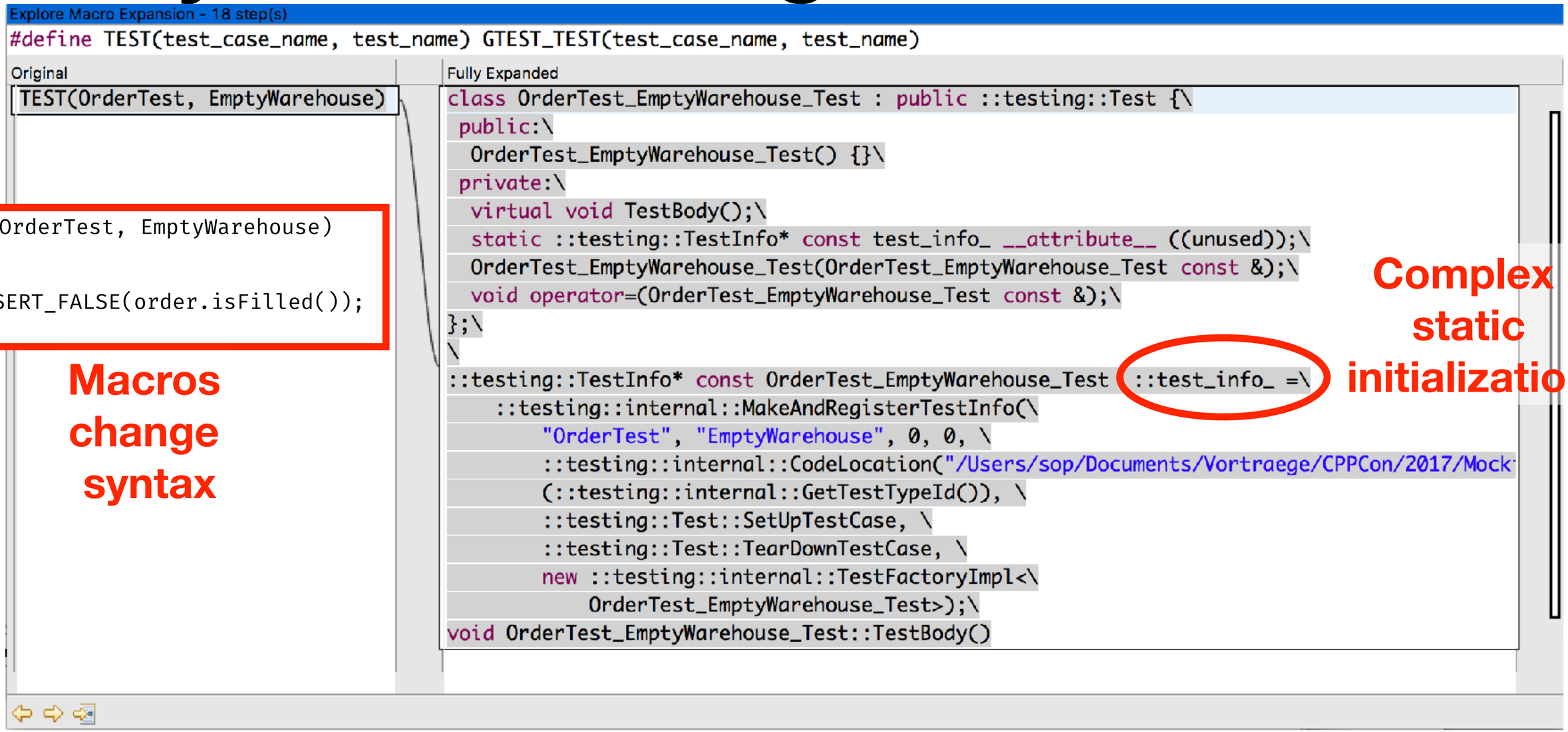
- he did call back:
  "I can not offer you a job,
  but you can have my job."

- I took over remains of Erich's C++ team and a server Application Framework
  - now called COAST: www.coast-project.org

1997-2002: *i*topia

- C++ Server Application Framework (Financial Industry)
- C++ Unit Testing and Test Automation (Test Infection)
- Delivered high-quality solutions, some still operational
- Agile Software Development and Wiki Wiki Webs work
- High-quality leads to better developers and retainment
- High-quality leads to cancelled maintenance contracts
- High-quality leads to fewer high-level client contacts

Why test-auto registration is bad



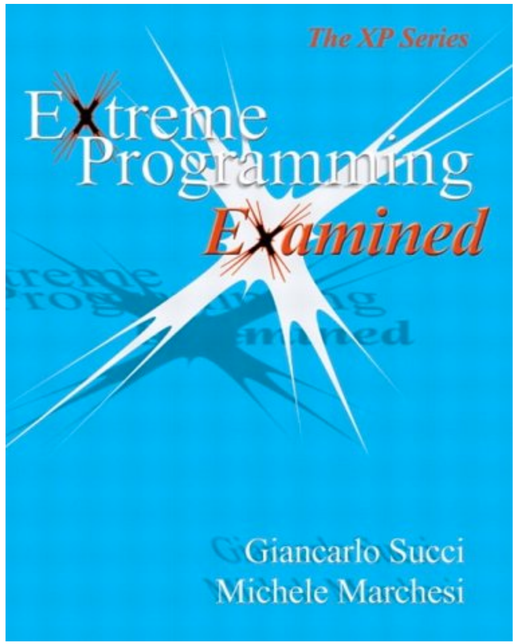
Lessons Learned

Lessons Learned

- To advance you need to take risks
- Do not be shy to use your network
- If you can not change your organization change your organization
- Leading a team and evolving it is gratifying
- Producing higher quality software than others might be bad for business
- Working software lives longer than ever intended

1998 - : Agile

Following Rituals is NOT Adhering to Values



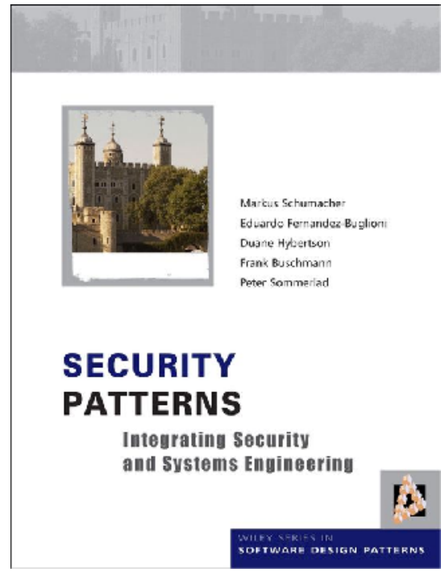
My most important Practices:

- **Unit Testing and TDD**
- **Refactoring and Simple Design**
- **Rhythm and Feedback**
- **Open honest Communication**

sometimes live takes you off-track

BMW at German F1GP year 2000 at Nürburgring

My Y2K Problem treatment and recovery



Lessons Learned

Lessons Learned

- **Be self-sufficient but accept and appreciate help**
- **As a patient you are responsible for yourself**
- **Be patient, take your time to heal**
- **Breaks are important**
- **Exercise your body, you only have one**

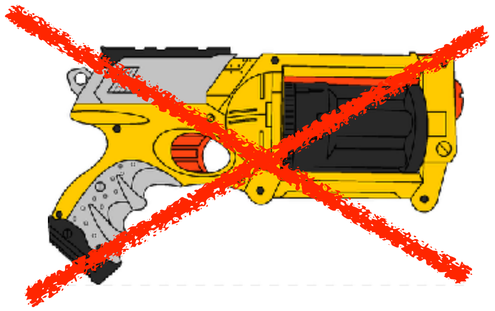
2004-20xx: HSR and IFS

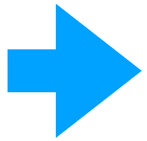
- Teaching (subjects over the years):
  - Agile and Pragmatic Software Engineering:
    i.e., Automatic Testing and Refactoring
  - C++ (more modern than I used to use it)
    see Moo/Koenig: Accellerated C++ and Andrescu: Modern C++
  - Concurrent and Network Programming (Threads and Sockets)
  - Patterns and Frameworks
- Student projects: IDE Tooling: Refactoring and Static Analysis
- Foundation of IFS Institute for Software

Lessons - Learned?

- Some students seem to have an ambivalent take on me: -->
- But I try to help them learn
- Asking a bit more than a student is comfortable with will actually help them to perform much better than one thinks they are capable
- Students have the right to fail

My Goal: Eliminate Bad Software!



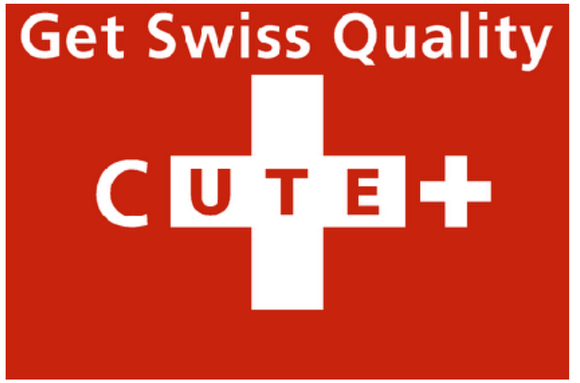


2005- Refactoring Tools

- Platform Eclipse Language ToolKit (LTK)
- From 2005 on earliest attempts with C++ Refactoring
- Ruby Refactoring (2007 - extinct)
- Python Refactoring/Type deduction static analysis (extinct)
- Javascript Refactoring (failed - too many different styles)
- Scala Refactoring Library (Mirko Stocker)
- Workshop on Refactoring Tools WRT2012
- IEEE Software Special Issue on Refactoring

"You are crazy"
"That is impossible!"

2005 - 2xxx: C++ IDE tooling



- CUTE: C++ Unit Testing Easier with Eclipse plug-in
- C++ Refactoring Infrastructure in Eclipse CDT
- C++ parser upgrades, e.g., const-evaluation for C++11 and 14
- Mockator: C++ seam introducing refactorings and simple mocking
- Includator: C++ header file clean up (was nick-named "REDuce HEAder Dependency" before)
- Templator: C++ template instantiation visualization
- Elevator: C++ code modernization: {init}, char* -> string, [] -> std::array,
- Constificator: const cleaner and east/west const reformatter
- SConsolidator: SCons integration
- CGLadiator: C++ Core Guideline Checkers and Quick-Fixes
- Conanator: Conan package manager intetration

Cevelop

Your C++ deserves it

Download IDE at:
www.cevelop.com

2005-2xxx: accu speaker

professionalism in programming

- Introduction to Security Patterns
- Understanding Security with Patterns,
  Only the Code tells the Truth
- Towards Simple Code (workshop),
  CUTE C++ Unit Testing Easier
- The Simplicity Workout,
  C++ Refactoring and TDD with Eclipse CDT
- Patterns of Simplicity,
  Design Patterns with Modern C++
- Agile C++ through Advanced IDE Features,
  Simple Software: Regain Control through Decremental Development
- C++11 for the rest of us: Simpler Code with More Power
- C++14 an Overview of the new standard for C++(11) programmers
- Compile-time computations in C++14
- Using Units, Quantities and Dimensions in C++14,
  Visualize Template Instantiations - Understand your Template Bugs
- C++ Core Guidelines - Modernize your C++ Code Base
- Mocking Frameworks considered, harmful?!

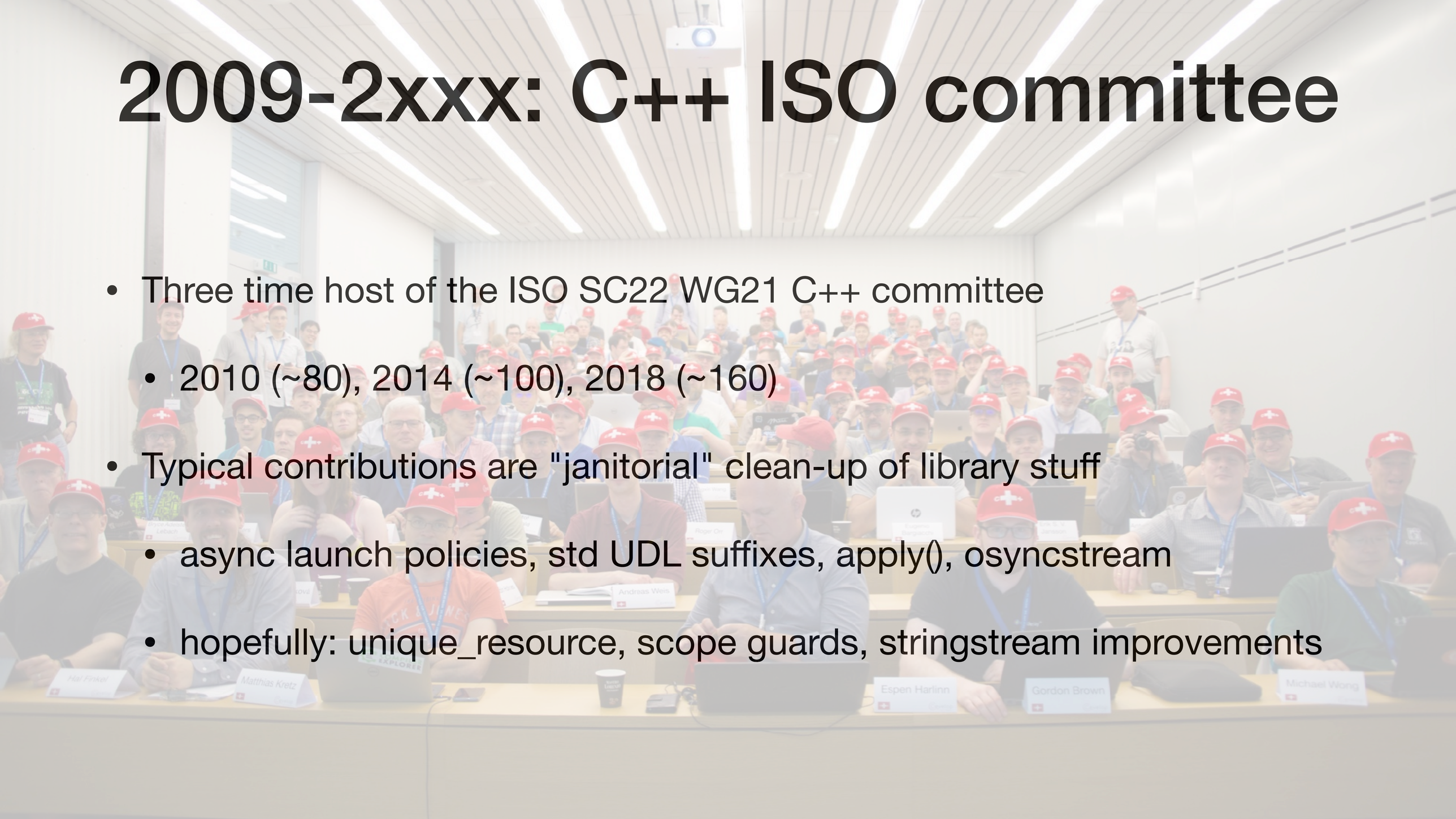
Less Code

=

More Software


Hal Finkel
Matthias Kretz
COMPILER EXPLORER
Andreas Weis
Roger Orr
Eugenio Bargiacchi
Erik S. V. Jansson
Espen Harlinn
Gordon Brown
Michael Wong

2009-2xxx: C++ ISO committee



- Three time host of the ISO SC22 WG21 C++ committee
  - 2010 (~80), 2014 (~100), 2018 (~160)
- Typical contributions are "janitorial" clean-up of library stuff
  - async launch policies, std UDL suffixes, apply(), osyncstream
  - hopefully: unique_resource, scope guards, stringstream improvements


C++ MEETING


Bleifrei 95
Bleifrei 95

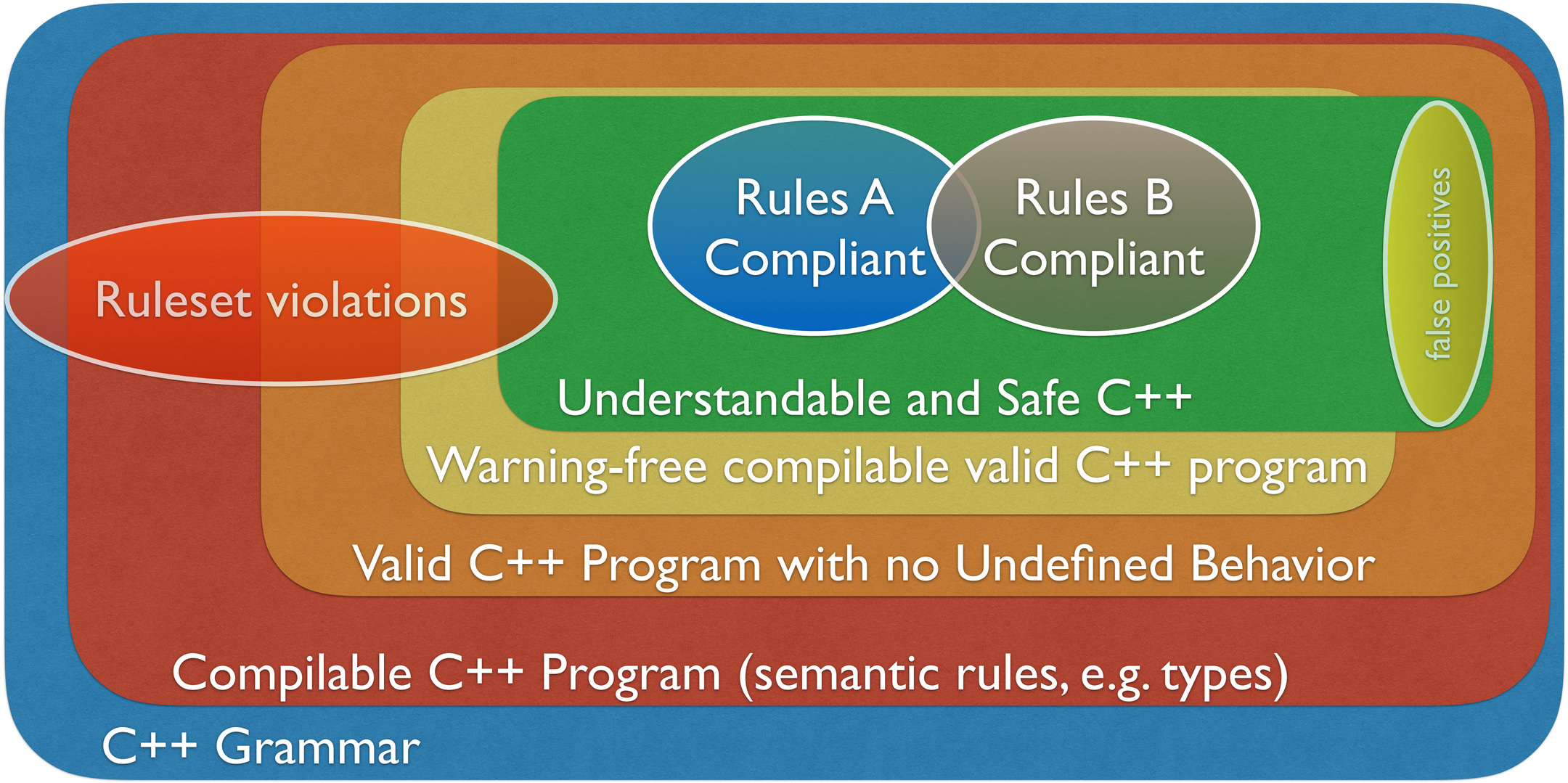
Lessons Learned

- You learn a lot, being in a room with brighter people than you
- Even small contributions can be valuable
- Keeping up-to-date with C++'s evolution is hard
- Nobody knows everything about C++, be humble
- Be patient, it can take years for a proposal to enter std

2017- : C++ Guidelines (again)

- Actually started earlier with C++ modernization plug-ins
  - refactor from char*, C-Arrays
  - refactor to C++11 initializers, east const west,
- Plug-ins for adhering to C++ Core Guidelines
  - Feedback and Remedy!
- I joined AUTOSAR C++ guidelines and MISRA C++ guidelines groups
  - So they do not repeat the MISRA C++ 2008 desaster

Safety Guidelines Goals: Limit Valid Programs



not to scale

Lessons Learned

Lessons Learned

- No set of guidelines is perfect and suits in all contexts
- Giving Feedback to Developers is important
- Immediate Feedback in Context is more helpful to learning
- Immediate Remedy appreciated, especially with "false positives"
- Developers are stuck with bad practices and non-IDEs
  - convincing them of a better way is daunting, but it was so in the 1990s
- An IDE can be more helpful than a compiler-based tool

My todays C++ Lessons for you

- Prefer Values and Regular Classes
- Wrap primitive types, especially in parameters
- NO raw Pointers, NO C-style Arrays, size() is not signed
- Be aware of potential dangling ([&] captures, string_view,...)
- Consider deleting move-assigment to make a type non-copyable
- East const is the right way! Sign http://slashslash.info/petition/ !

Use unique_ptr for heap objects !

```
gsl::owner<T>
```

```
T *
```

```
char const * const
```

```
T[]
```

```
std::shared_ptr<T>
```

```
std::unique_ptr<T>
```

```
std::experimental::observer_ptr<T>
```

```
std::array<T,N>&
```

```
T &
```

```
T const &
```

```
std::reference_wrapper<T>
```

Employ C++17 Aggregate Extensions

```
template <typename V, typename TAG>
struct strong {
  static_assert(std::is_object_v<V>,"must keep real values");
  using value_type=V;
  V val;
};
```

```
struct Size: strong<unsigned,Size>,ops<Size,Eq,Cmp,Inc,Add,Out> {
};
static_assert(sizeof(Size)==sizeof(unsigned),"no overhead");

void testSizeworks(){
  Size sz{42};
  //ASSERT_EQUAL(42,sz);// doesn't compile
  //ASSERT_EQUAL(42u,sz);//doesn't compile
  ASSERT_EQUAL(Size{21}+Size{21},sz);
}
```

Polymorphic Base Class

```
struct drawable {
  virtual ~drawable()=default;
  virtual void draw(screen& on)=0;
protected:
  drawable& operator=(drawable&&)noexcept=delete; // prohibit move and copy
};
using widget=std::unique_ptr<drawable>;
using widgets=std::vector<widget>;
```

with deleted move assignment

Making a class non-copyable

What you get

What you write	default constructor	destructor	copy constructor	copy assignment	move constructor	move assignment
nothing	defaulted	defaulted	defaulted	defaulted	defaulted	defaulted
any constructor	not declared	defaulted	defaulted	defaulted	defaulted	defaulted
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	defaulted
destructor	defaulted	user declared	defaulted (!)	defaulted (!)	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted (!)	not declared	not declared
copy assignment	defaulted	defaulted	defaulted (!)	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	defaulted	deleted	deleted	not declared	user declared

Howard Hinnant's Table: https://accu.org/content/conf2014/Howard_Hinnant_Accu_2014.pdf
Note: Getting the defaulted special members denoted with a (!) is a bug in the standard.


SZ·69833
HEIDEGGER

I am looking forward to my next 35 years of programming

peter.sommerlad@hsr.ch
@PeterSommerlad
www.cevelop.com