Software Product Lines
Part 1: Introduction and Overview

Daniel Strüber, Radboud University

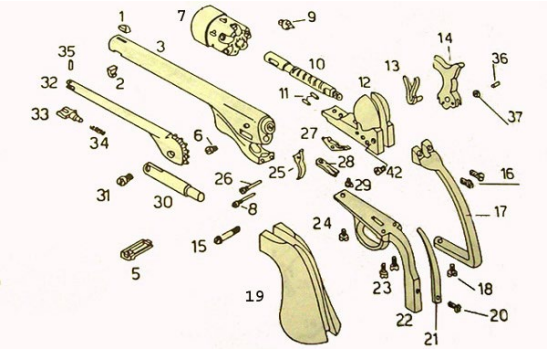
with courtesy of: **Sven Apel, Christian Kästner, Gunter Saake**

Agenda

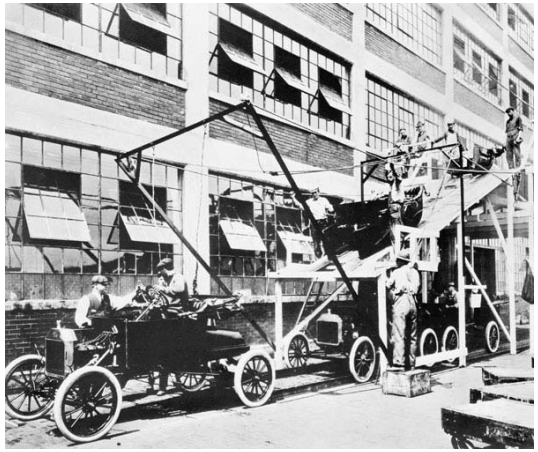
- Formal and organizational matters
- A short introduction to the topic
  - Product lines in industry
  - Software product lines
  - Challenges to software product lines

Background

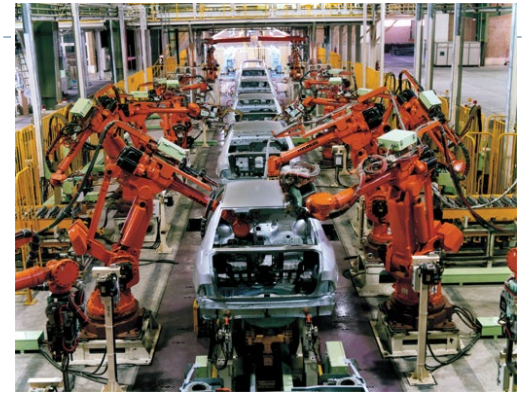
Industrial revolution



1826
exchangeable parts
(John Hall, after 25 years of trying)



1901
conveyor belt
(Ransom Olds/Henry Ford)

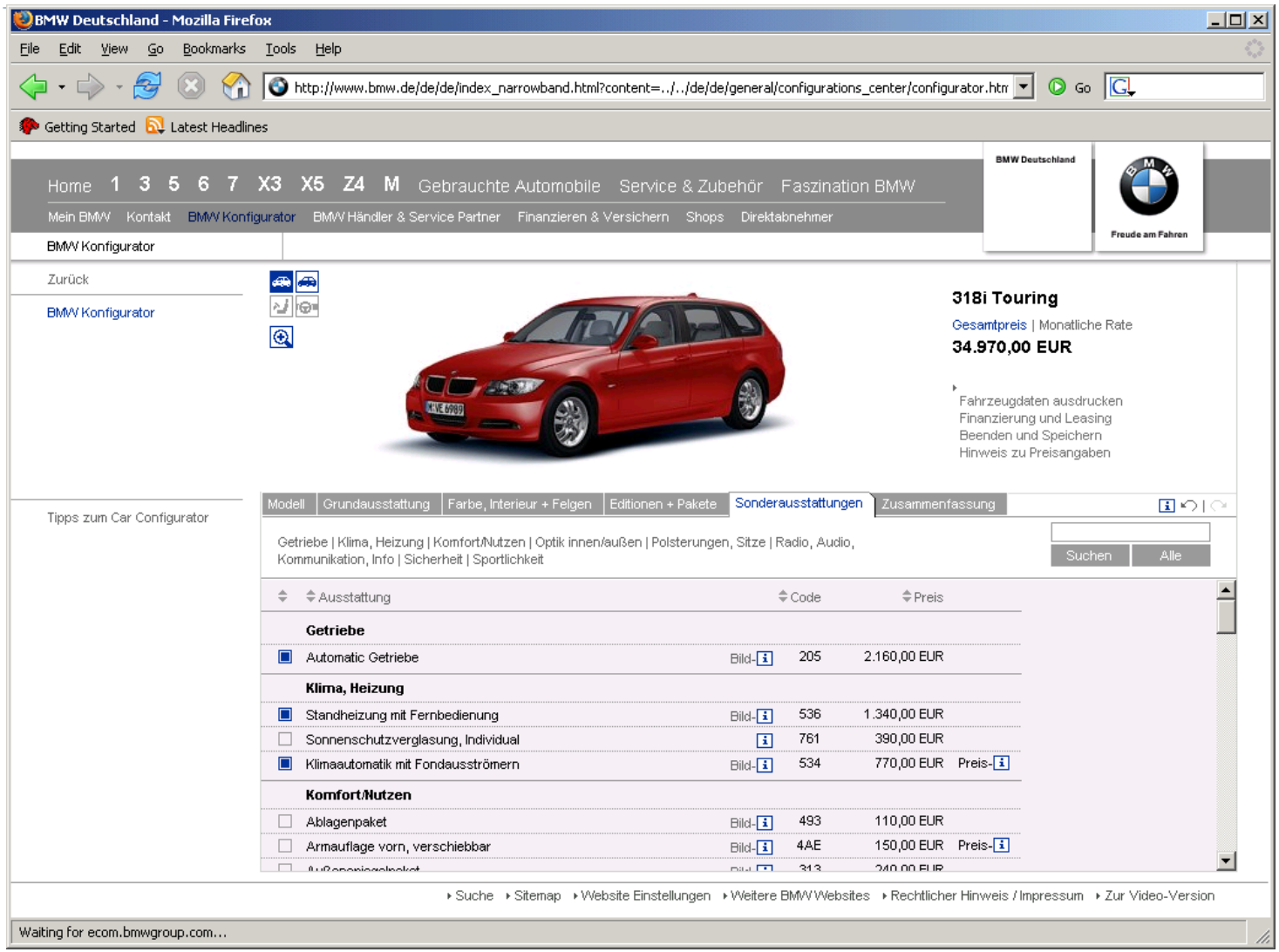


1980s
automated production line
(first industrial robot 1961
at General Motors)

[Czarnecki & Eisenecker 2000]

Product lines in industry

Car product line



Diversity of variants

Example: BMW X3

Ceiling: **90.000** variant options	Car doors: **3.000** variant options	Rear axle: **324** variant options

„Variants are a significant leverage for our operating results.“
— **Franz Decker**, head of variant management programme, BMW Group

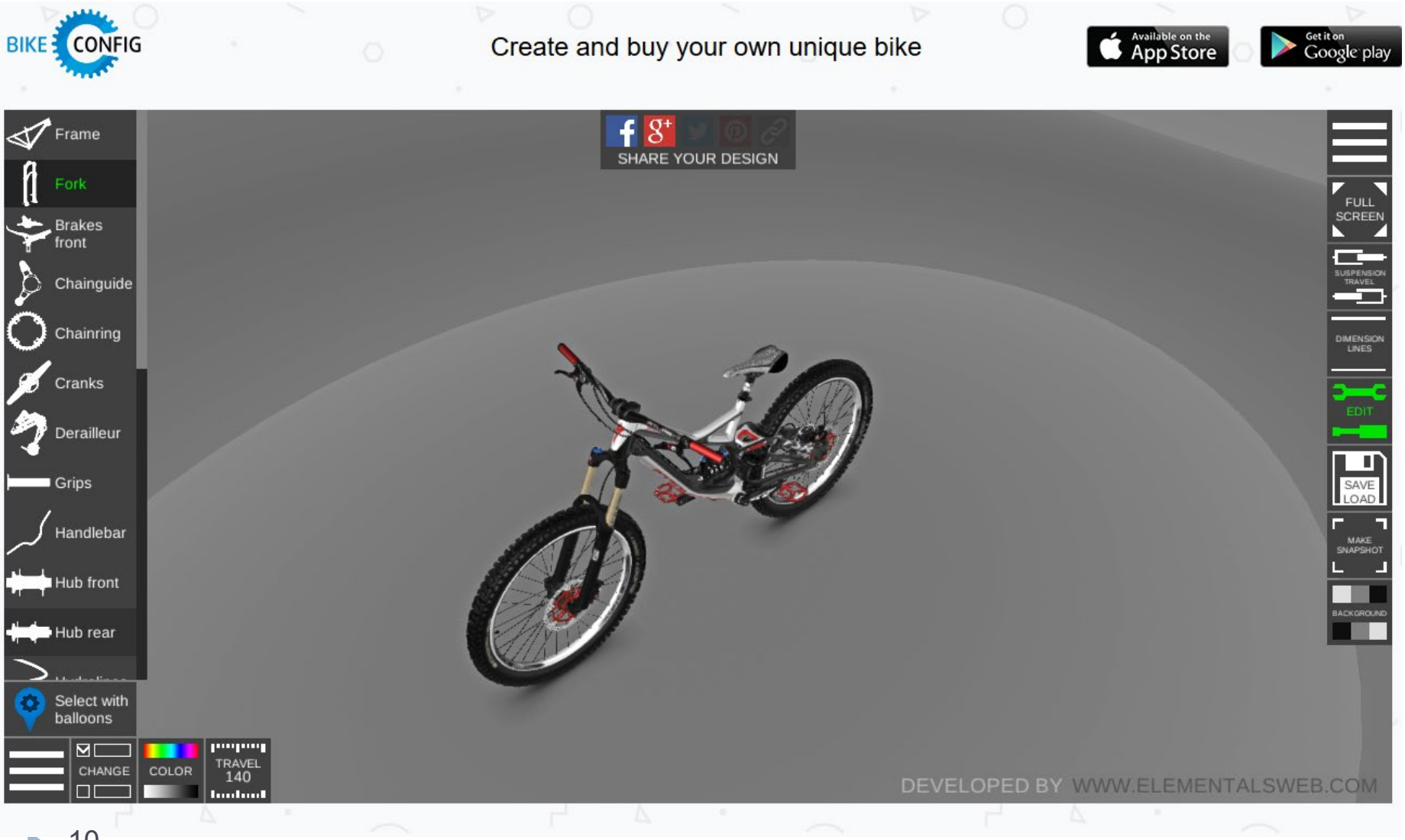
Car product lines 20 years ago

- Choice was restricted to models and a few extras, like roof-mounted luggage rack or alternative CD player
- One variant (Audi 80, 1.3l, 55PS) = 40% of sales

Car product lines today

- 10^{20} possible variants of an Audi
  10^{32} possible variants of a BMW
- Hardly a car leaves the production plant with the same features as the one before it
- 100 different floor units for one model, based on engine and equipment
- 50 different steering wheels (3 vs. 4 spokes, wood vs. plastic vs. leather, heating, colors)

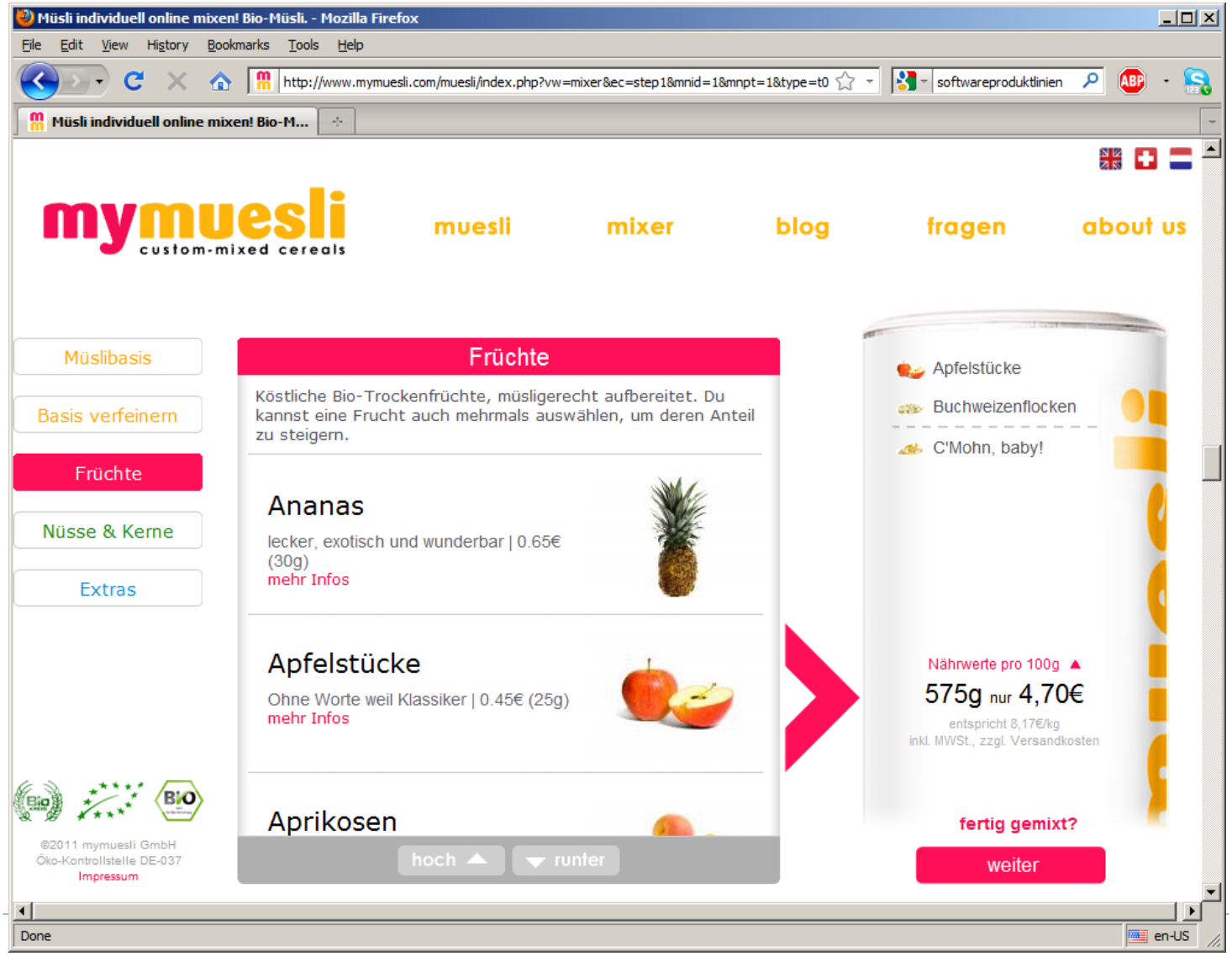
Not just cars….



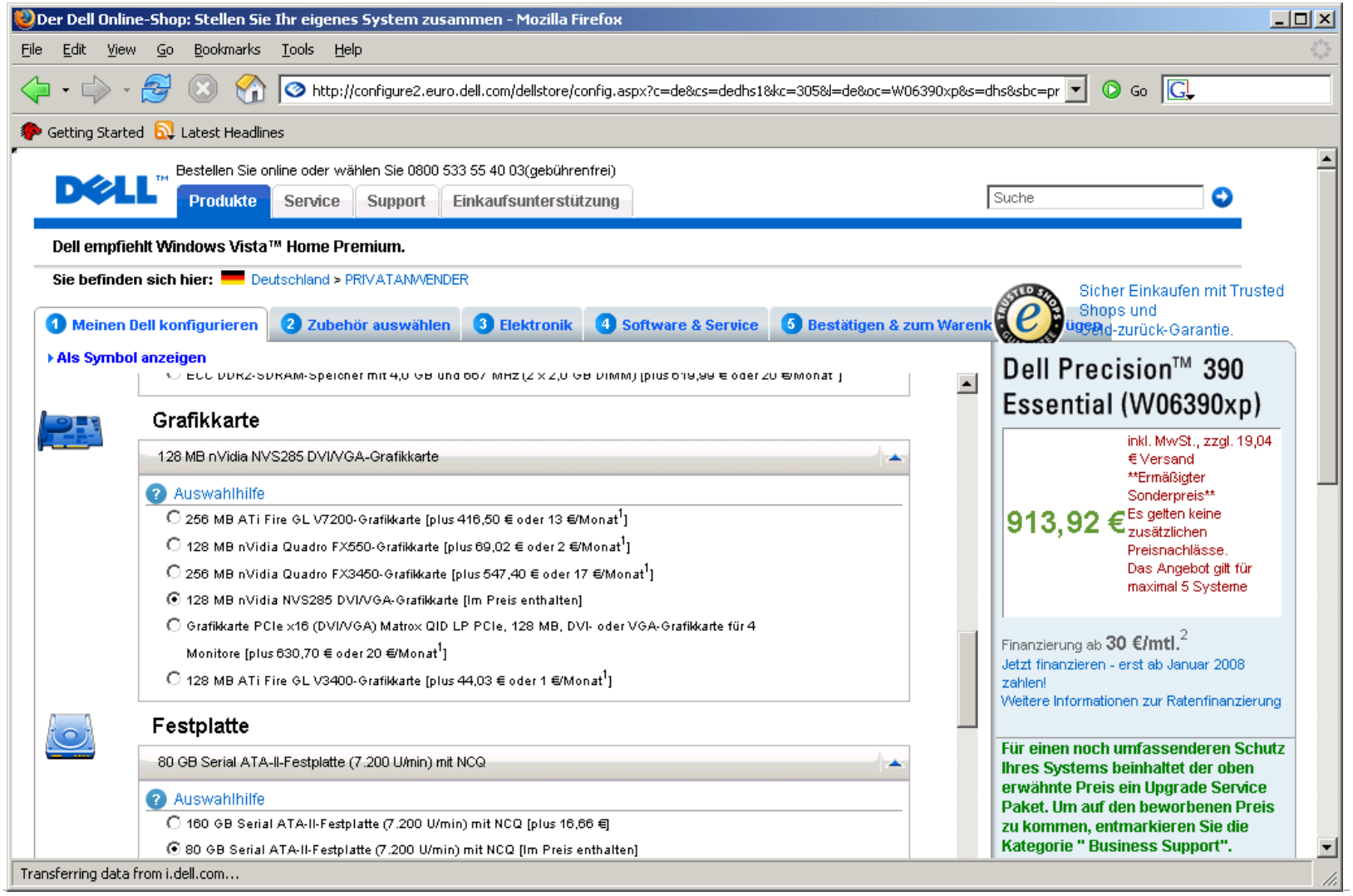
More product lines

More product lines

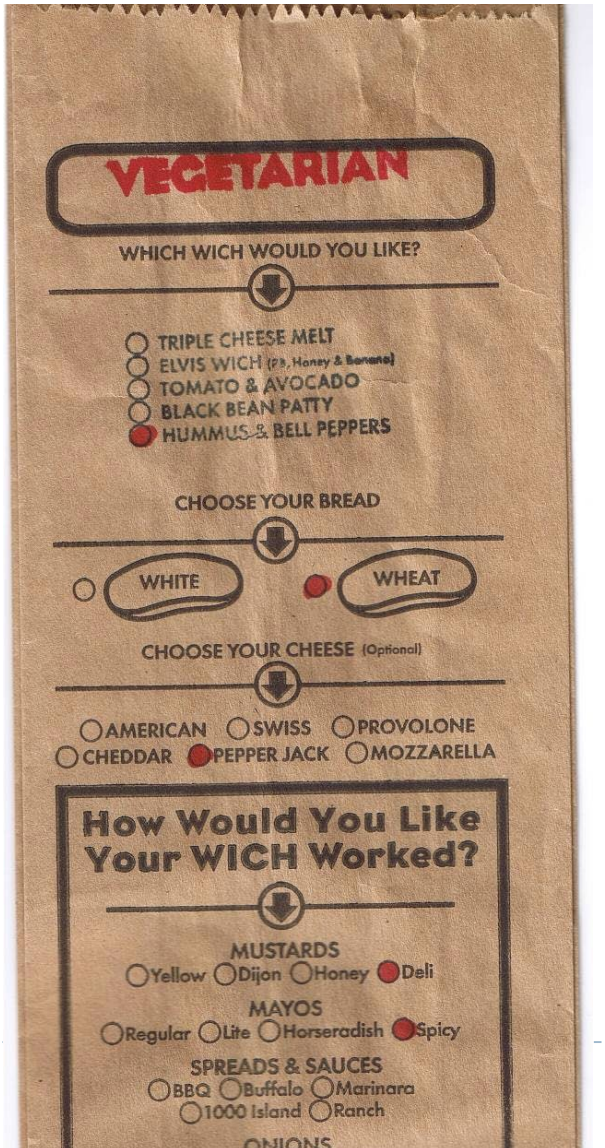
More product lines

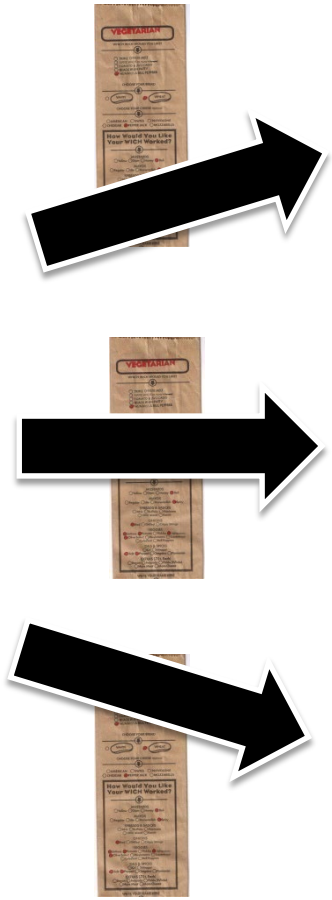


More product lines



More product lines




VEGETARIAN
How Would You Like Your WICH Worked?
VEGETARIAN
How Would You Like Your WICH Worked?

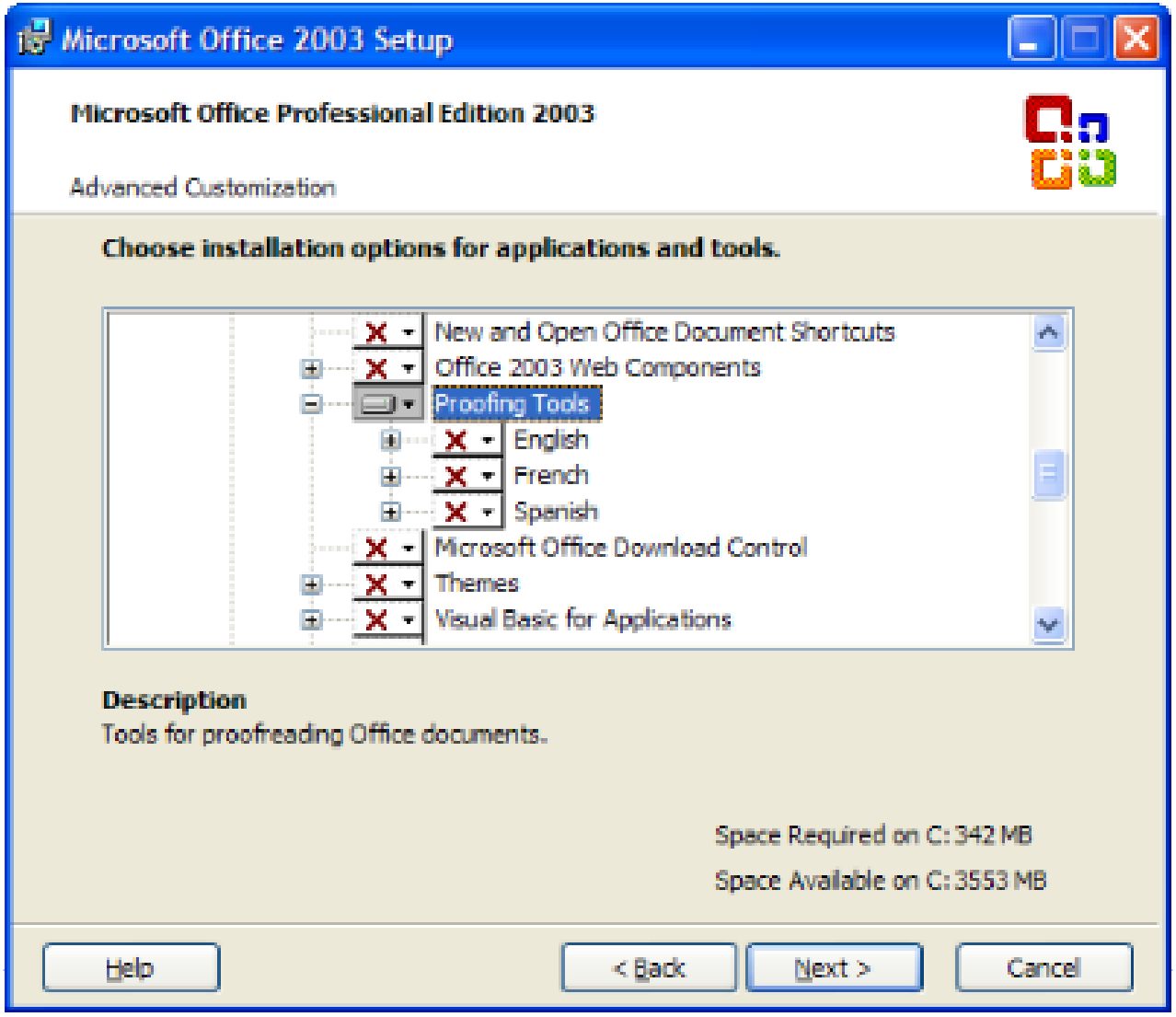
And Software?

- Modern application software systems are Swiss army knives
  - Bsp.: Windows 10, Open Office, Oracle, SAP myERP, Adobe Photoshop, AutoCAD
- Specialized software und systems for embedded systems more and more important
  - Bsp.: phones, sensor networks, microwaves, TVs, weather stations, cars, chip cards, on-board computers, routers, ubiqutious computing
  - 98% of computing systems in use are embedded systems
  - Resource constraints and heterogenous hardware require custom-tailored solutions
  - Frequent from-scratch development, long time-to-market, high development costs

Why custom-tailored software?

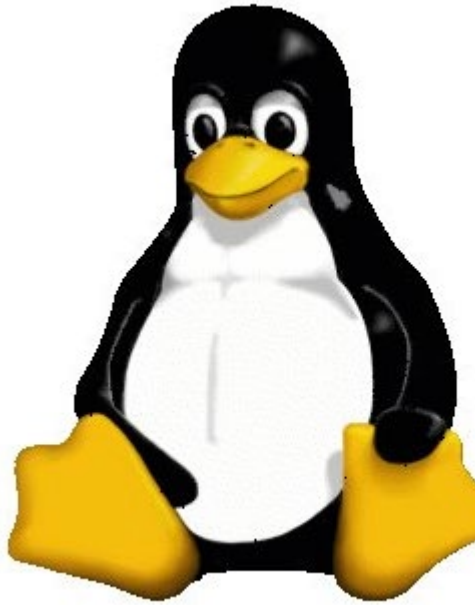
- Personalization
  - Individual requirements, Look-and-Feel, special algorithms, law settings, hardware, …
- Resource constraints
  - Energy consumption, performance, memory use, ….
- Software and product quality
  - Usability
  - Unused functionality as a risk
  - Maintenance and testing effort grows with amount of functionality
- Costs, effort, flexibility, competition…

Features in Microsoft Office



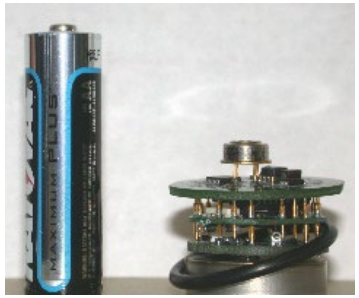
Linux kernel

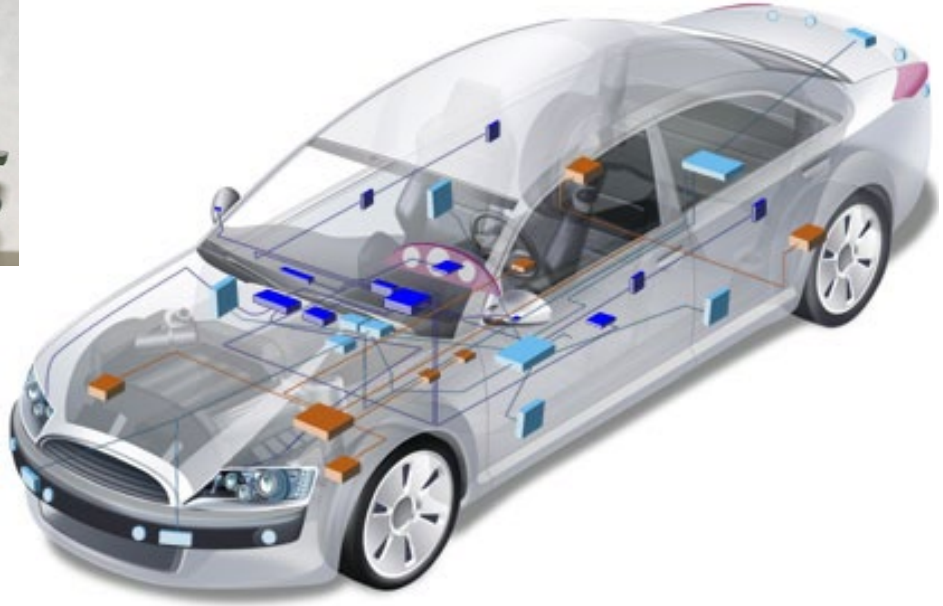
- about 6.000.000 lines of source code
- highly configurable
  - > 10.000 configuration options! (x86, 64bit, …)
  - almost all source code is „optional“



Databases

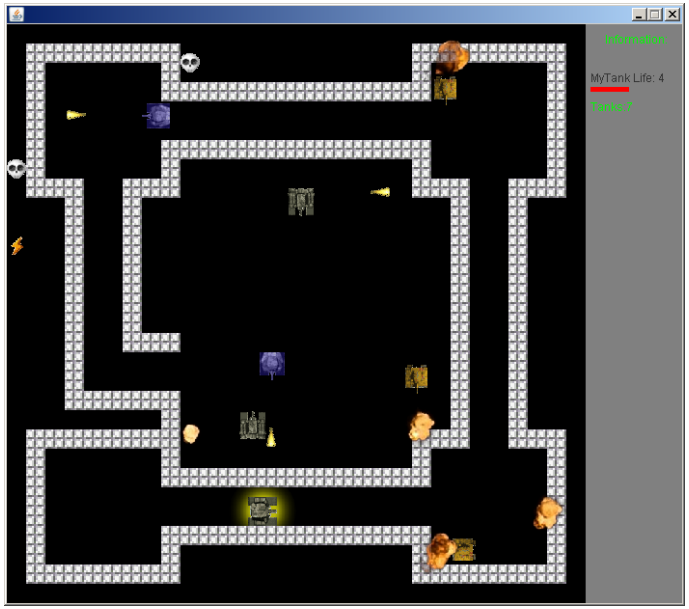
- Continously increasing amounts of data
- Often embedded into contexts with resource constraints

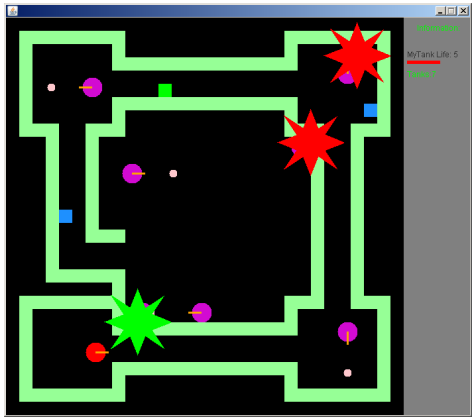




Printer firmware

Games


Information
MyTank Life: 4
Tanks 7


Information
MyTank Life: 5
Tanks 7

Software product lines in industry

- HP: printer drivers/firmeware
- Nokia: phone software, browser
- Phillips: high-end TVs, medical technology, …
- TomTom: embedded navigation systems
- Cummins: Diesel engine control software
- LG: lift control software
- Ericsson: telecommunication switches
- General Motors: powertrains
- Many more: gas turbines, train control, ship control, frequency converter, internet payment gateway, helicopter avionics software, …

Goal of course

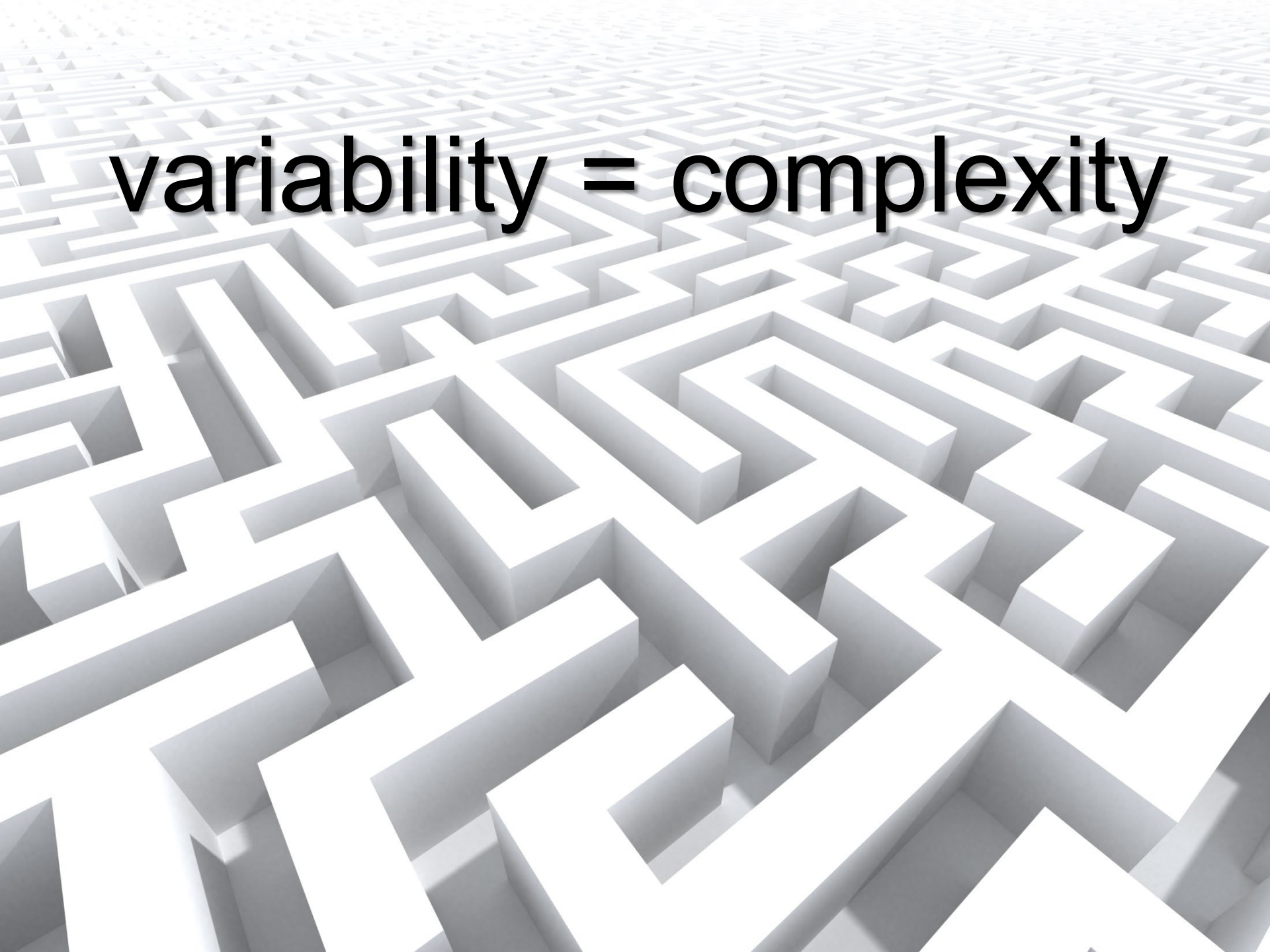
- Techniques for developing custom-tailored software that contains exactly the required functionality
  - Variants for different use-cases
  - Can easily add new variants, fast further development, reuse of proven functionality
  - Customer-specific production, specialization
  - Customization to available resources

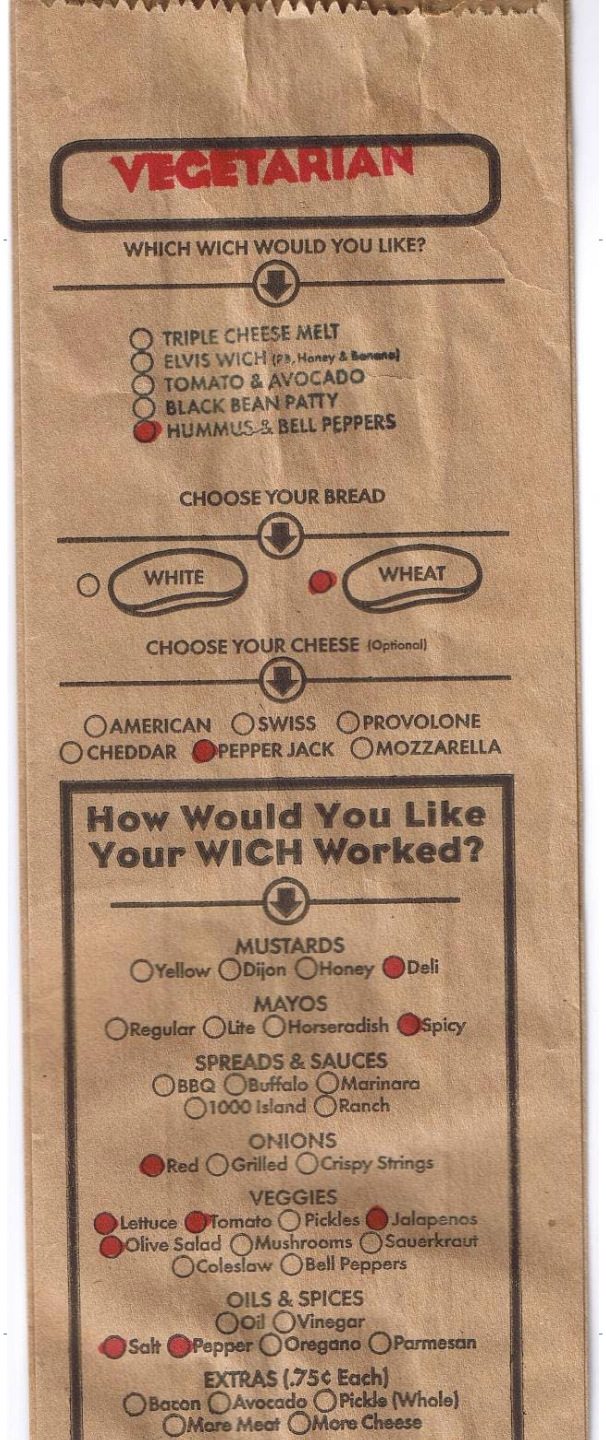
- Software product lines (SPLs):
  Configuration by selection of features (Dutch: kenmerken)

Estimation

about 80% of all software systems are software product lines or can benefit from software product line technology

Challenges

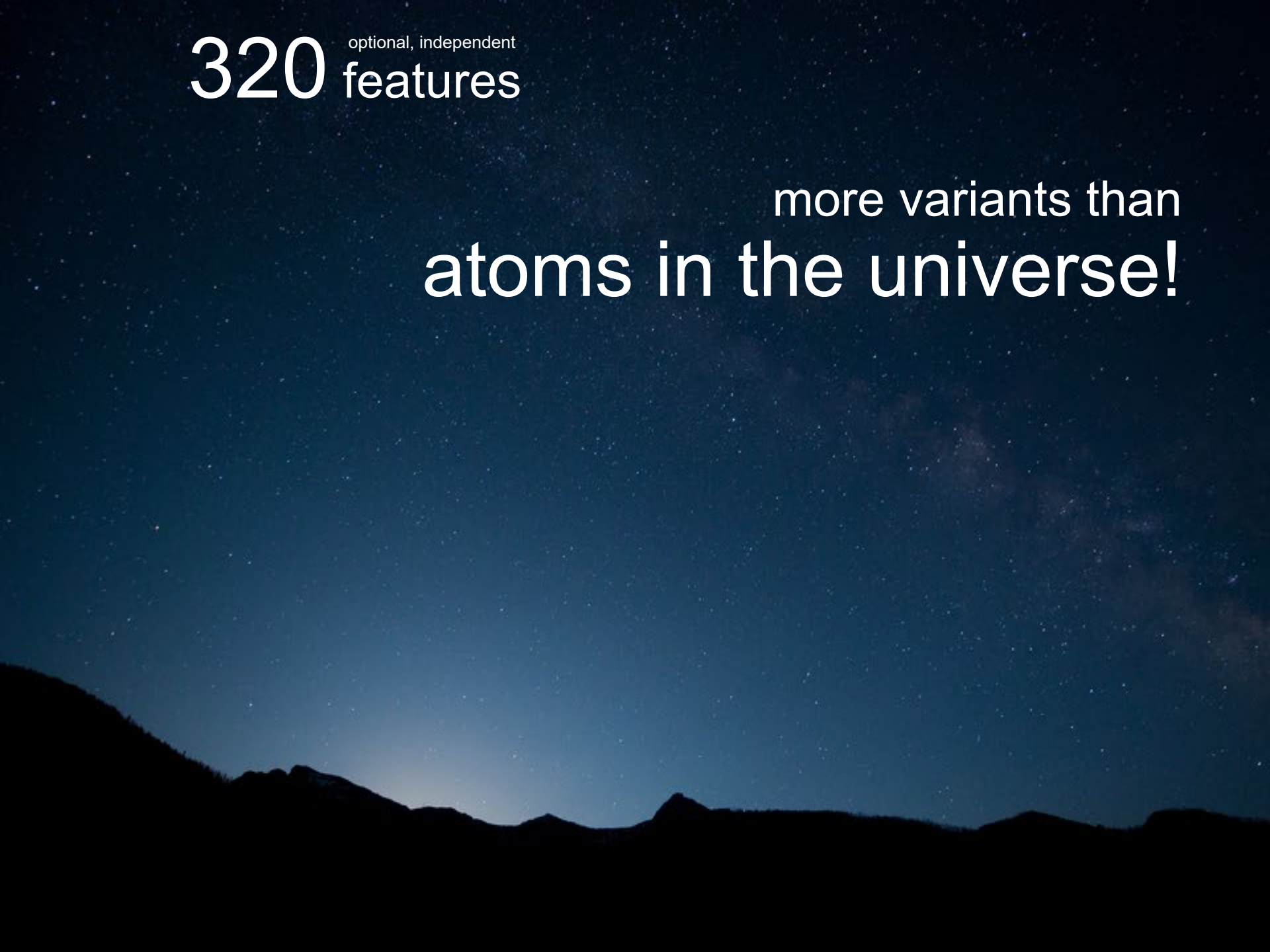

variability = complexity


VEGETARIAN
WHICH WICH WOULD YOU LIKE?
TRIPLE CHEESE MELT
ELVIS WICH (PB, Honey & Banana)
TOMATO & AVOCADO
BLACK BEAN PATTY
HUMMUS & BELL PEPPERS
CHOOSE YOUR BREAD
WHITE
WHEAT
CHOOSE YOUR CHEESE (Optional)
AMERICAN SWISS PROVOLONE
CHEDDAR PEPPER JACK MOZZARELLA
How Would You Like Your WICH Worked?
MUSTARDS
Yellow Dijon Honey Deli
MAYOS
Regular Lite Horseradish Spicy
SPREADS & SAUCES
BBQ Buffalo Marinara
1000 Island Ranch
ONIONS
Red Grilled Crispy Strings
VEGGIES
Lettuce Tomato Pickles Jalapenos
Olive Salad Mushrooms Sauerkraut
Coleslaw Bell Peppers
OILS & SPICES
Oil Vinegar
Salt Pepper Oregano Parmesan
EXTRAS (.75¢ Each)
Bacon Avocado Pickle (Whole)
More Meat More Cheese

33 optional, independent features

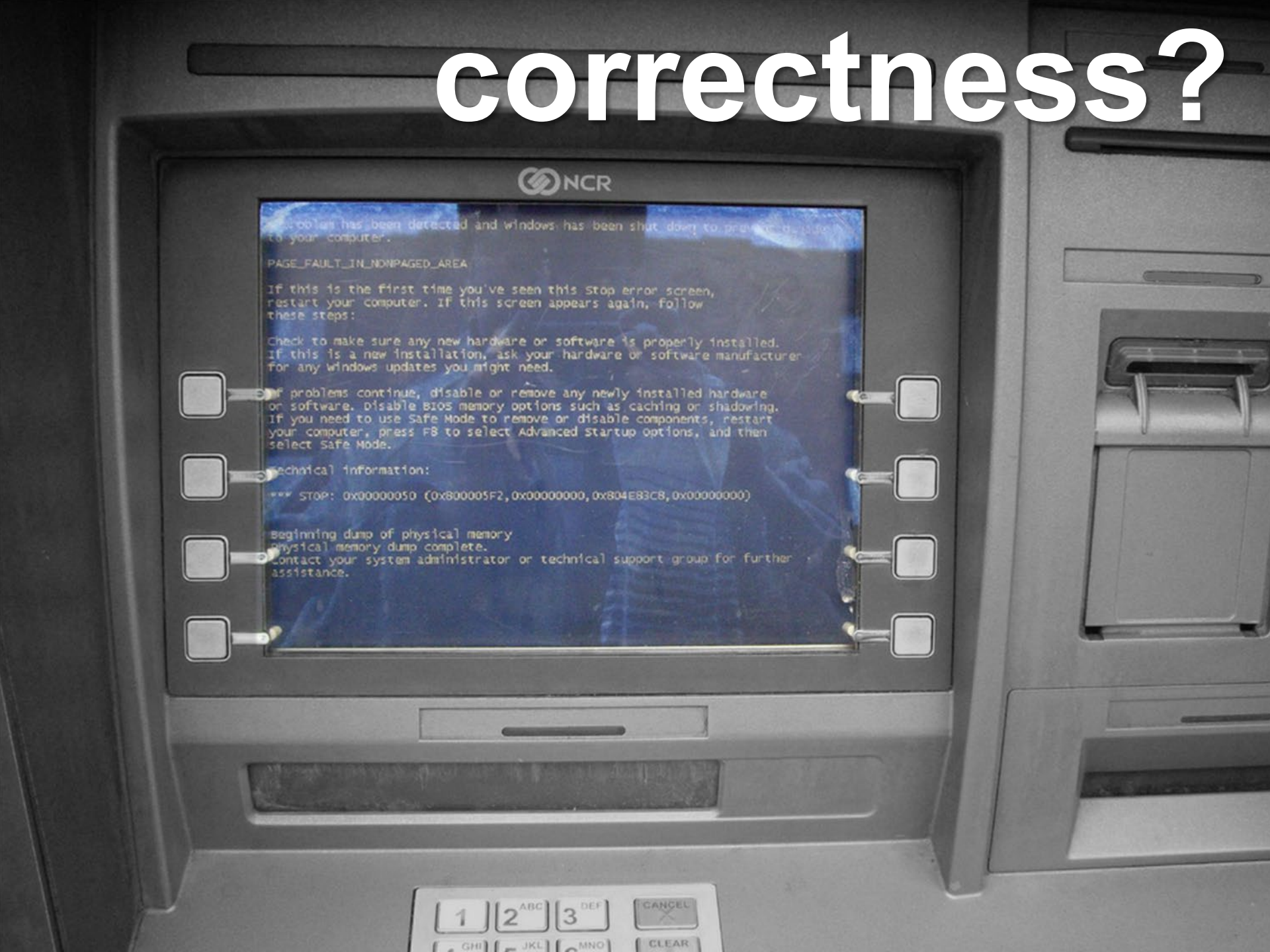
one custom-tailored variant for

every human on the planet

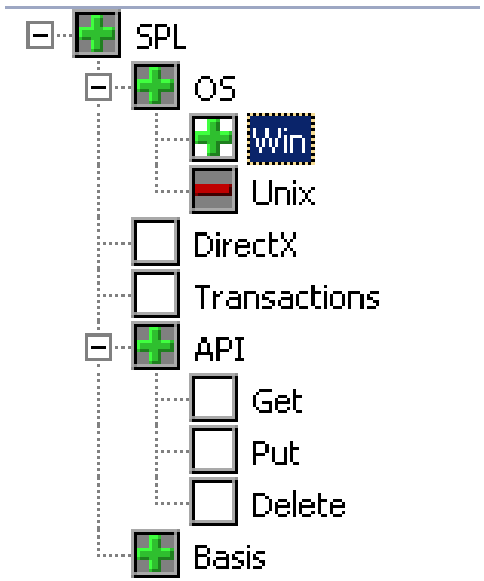

320 optional, independent features
more variants than
atoms in the universe!

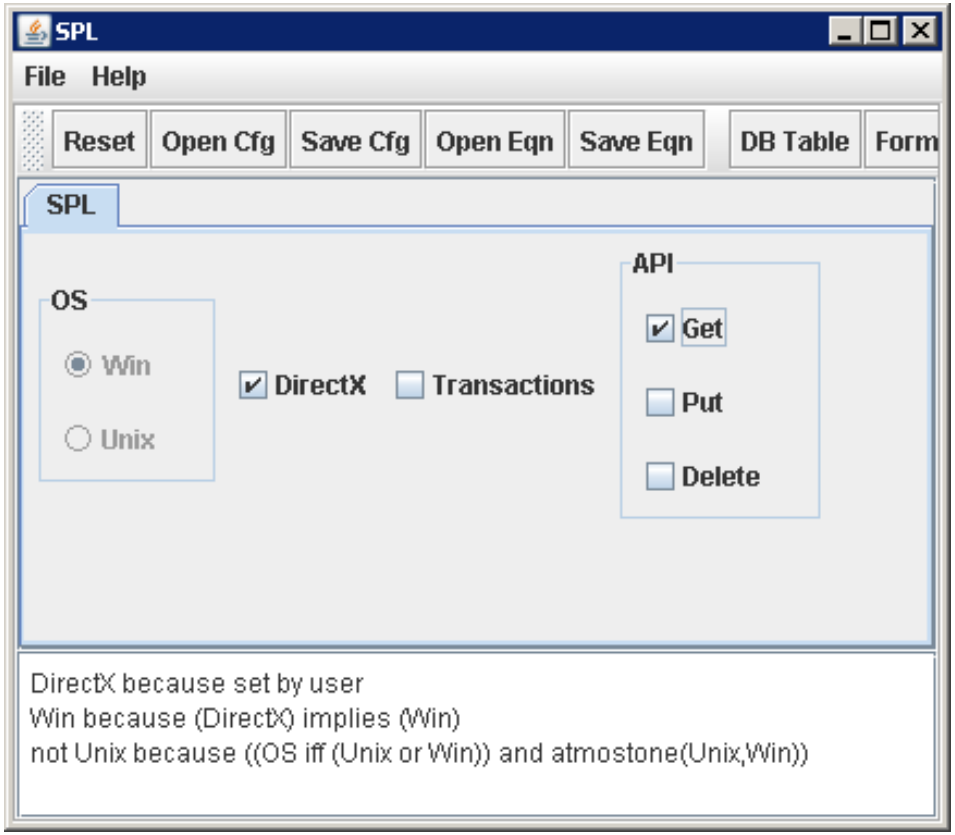
2000 features

10000 features

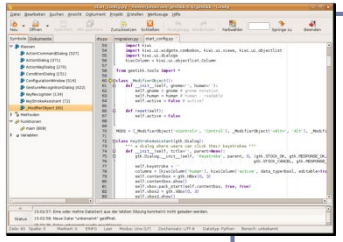

correctness?
NCR
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0x800005F2,0x00000000,0x804E83C8,0x00000000)
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
1
2 ABC
3 DEF
CANCEL
CLEAR

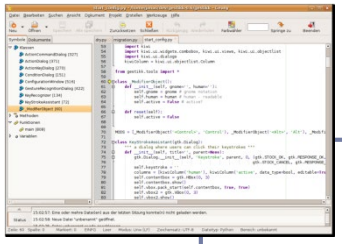
are all combinations useful?

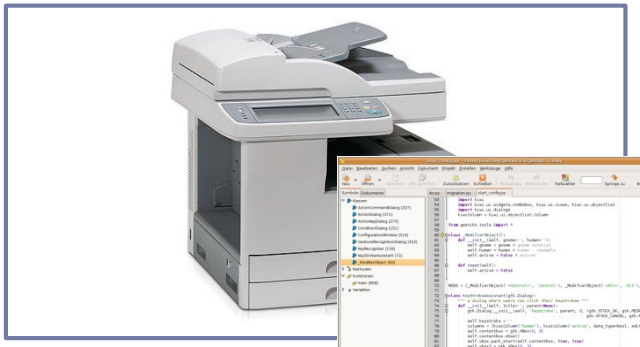


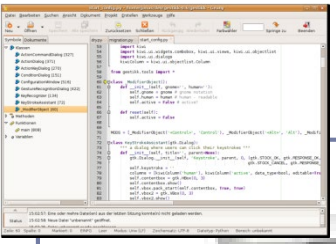


Reuse during implementation?

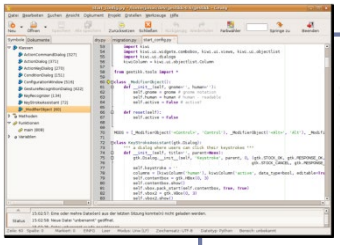








Where correct errors?



Idee: Systematic development process

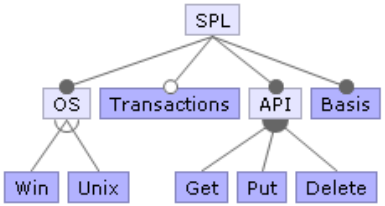
- Developing new variants from scratch is not economically feasible
- Creating variants by „clone and own“ is a possibility, but comes with many problems in practice
- Therefore: Custom-tailored software based on software product lines
  - From reusable parts
  - That can have alternative implementations
  - Customizable for specialized use-cases
  - Even usable in presence of extreme resource constraints

Design and implementation of features

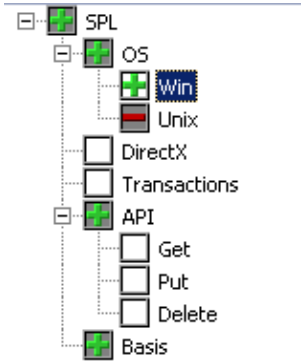
Domain Eng.

Application Eng.

Feature model

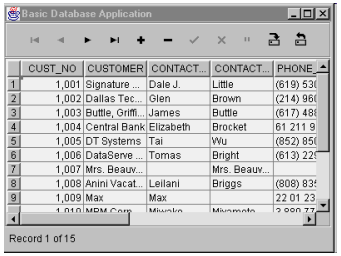


Reusable implementation artifacts



Feature selection

Generator



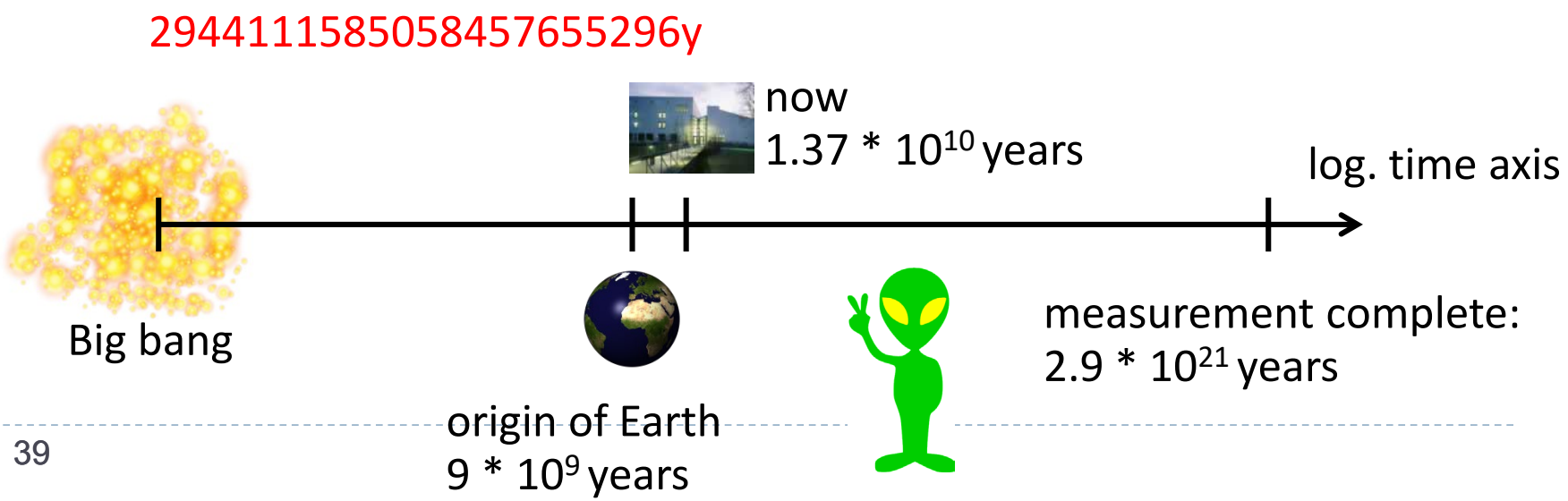
Final program

Non-functional properties

- Constraints and optimisation
- Optimisation for low energy consumption
- Optimisation for high performance
- Optimsiation for low footprint
- Special implementation
  (e.g., no RAM; few writing accesses)
- Security

Optimisation of non-functional properties

- SQLite ist an embedded and customizable system
  - Used in more than 500 million systems
  - Configurable with 88 compilation options
- Assuming we experiment with every SQLite variant:
  - 2^{88} variants and 5 min. per experiment (compile + benchmark)
    = 2^{88} * 5min / 60 (per h) / 24 (per d) / 365 (per y) =



Implementation: State of the Art

- If variant management is done, then mostly with #ifdef, templates, make, Git
- Example: Berkeley DB (mutex_int.h)

```
#ifndef _DB_MUTEX_INT_H_
#define         _DB_MUTEX_INT_H_

#ifdef HAVE_MUTEX_PTHREADS
#include <pthread.h>

#define         MUTEX_FIELDS
                pthread_mutex_t mutex;          /* Mutex. */
                pthread_cond_t  cond;           /* Condition variable. */
#endif

#ifdef HAVE_MUTEX_UI_THREADS
#include <thread.h>
#endif

#ifdef HAVE_MUTEX_SOLARIS_LWP
#include <synch.h>

#define         MUTEX_FIELDS
                lwp_mutex_t mutex;              /* Mutex. */
                lwp_cond_t cond;                /* Condition variable. */
#endif

#ifdef HAVE_MUTEX_UI_THREADS
#include <thread.h>
#include <synch.h>

#define         MUTEX_FIELDS
                mutex_t mutex;                  /* Mutex. */
                cond_t  cond;                   /* Condition variable. */
#endif

#ifdef HAVE_MUTEX_AIX_CHECK_LOCK
#include <sys/atomic_op.h>
typedef int tsl_t;

#ifdef LOAD_ACTUAL_MUTEX_CODE
#define         MUTEX_INIT(x) 0
#define         MUTEX_SET(x) (!_check_lock(x, 0, 1))
#define         MUTEX_UNSET(x)                  _clear_lock(x, 0)
#endif
#endif …
```

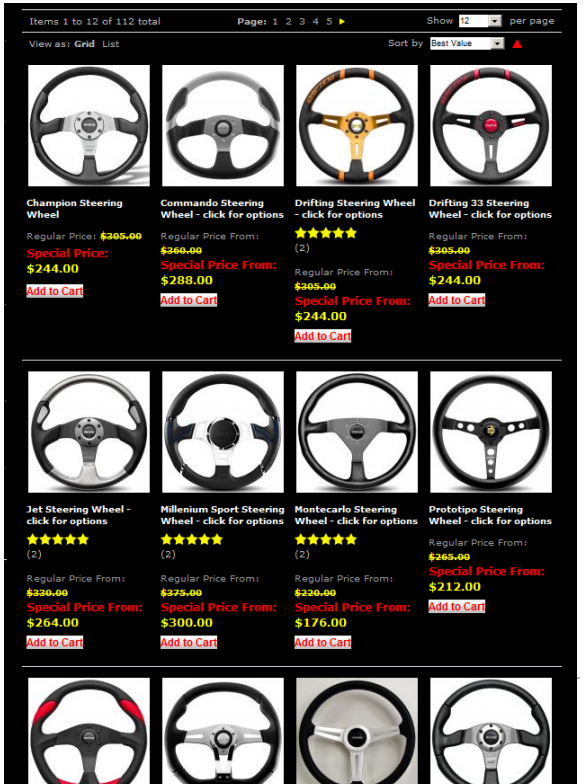
Implementation and maintainability?

```
class Stack {
void push(Object o
#ifdef SYNC
, Transaction txn
#endif
) {
    if (o==null
#ifdef SYNC
        || txn==null
#endif
        ) return;
#ifdef SYNC
    Lock l=txn.lock(o);
#endif
    elementData[size++] = o;
#ifdef SYNC
    l.unlock();
#endif
    fireStackChanged();
}
}
```

SPL implementation

- Different methods for implementing SPLs
- In this course we consider, t.ex.:
  - Versioning systems
  - Preprocessors
  - Components
  - Frameworks/plugins
  - Feature-oriented programming
  - Aspect-oriented programming
- …and further important topics
  - Feature interactions
  - Analysis and testing of software product lines
  - Virtual separation of concerns

What‘s a reasonable amount of variability?



Development cost
Investment cost
Maintenance cost
Logistical cost
Product cost (small quantities)

Focus

- Goal: Custom-tailored software
- Central concept: Features
- Focus on implementation of features
- Much source code, diverse paradigms and language extensions