CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Lecture 1: Complexity and Software Product Lines

Gregory Gay
(Portions of the slides by Thorsten Berger)
TDA 594/DIT 593 - November 2, 2021

Covid-19 Safety Procedures

- We will try to be flexible!
  - E-mail ggay@chalmers.se if you have concerns.
- Lectures
  - In-person or streamed (see Canvas).
  - We will monitor chat in livestream for questions.
  - ***STAY HOME*** if you are sick (with anything)!
  - **Maintain distance** in classroom if possible.

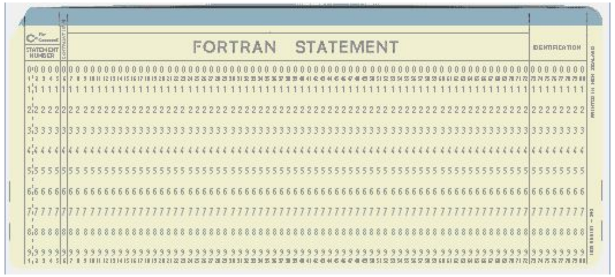
Covid-19 Safety Procedures

- Supervision and Group Work
  - Can conduct supervision sessions online or in-person.
  - Be flexible in how you meet with other students.
  - Do NOT meet your group if you have any symptoms.
- Vaccination
  - Seriously. Get vaccinated. Come on.
  - Kry have walk-in clinics on campus.

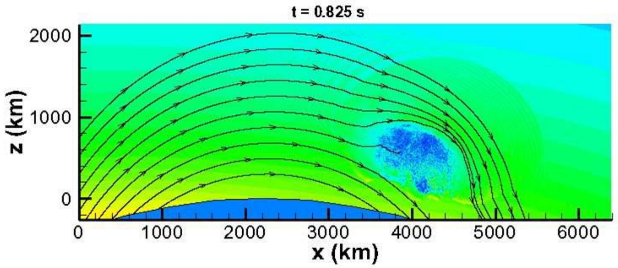
In the beginning...

Software Was Small

- Both physically…



- And in scope.



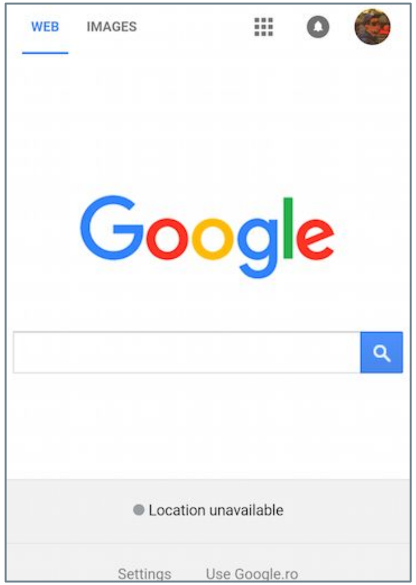
Software Starts to Grow Up

- Languages like C introduce file linking.
  - Enables organization of code and reuse of code.
- SIMULA-67, Smalltalk introduce objects.
  - Enables organization of code into focused units that work with other objects to perform larger tasks.
    - Sections of the code “activate” when needed.
    - We can group together related functionality, ignore unrelated functionality, and find what we need when making changes.
    - Code can be reused in future projects.

Fast forward to the present day...

Our Society Depends on Software

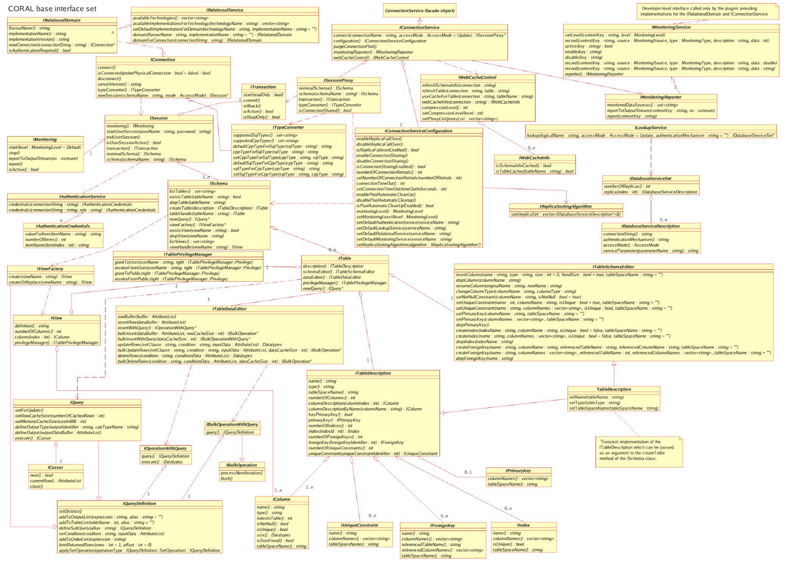
This is software:



So is this:

Under the Hood

- Systems have millions of lines of code.
  - In hundreds of classes.
- We want to reuse code.
  - On different hardware.
  - In many different apps.
- We want the systems to live for years.



Growing Pains

- Hard to understand complex systems.
- Hard to find the code you need.
- Hard to plan for new hardware, new variants, ...
- Results in chaos.
  - Only 16.1% of projects delivered on time.
  - 31.1% cancelled before delivery.
  - Delivered projects may be flawed.

Designing Modern Software

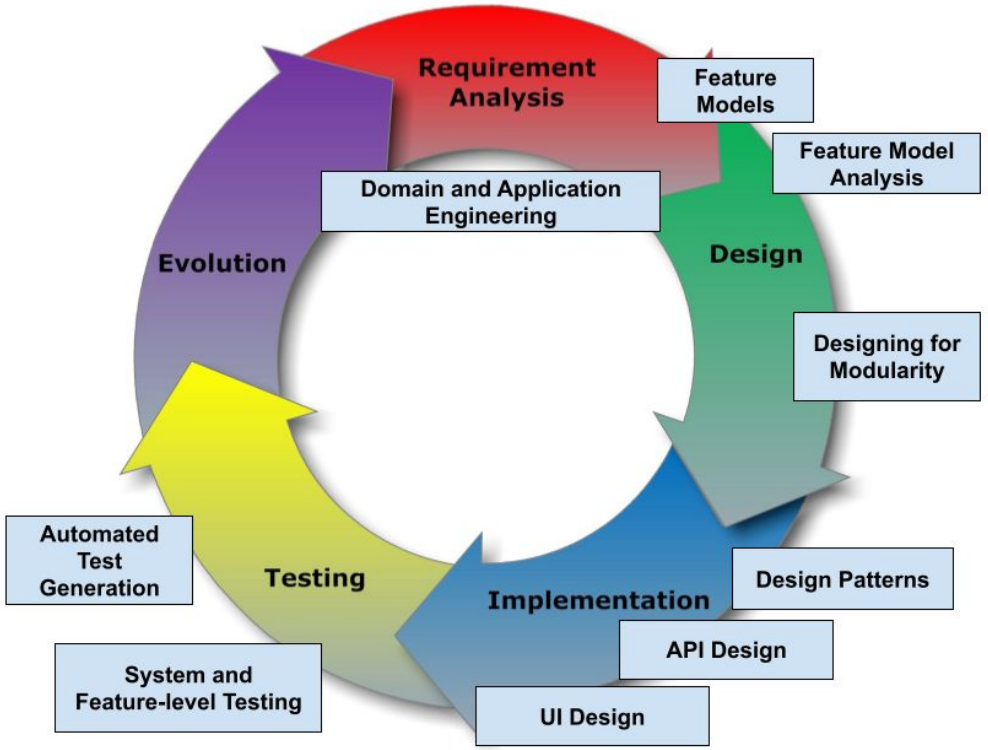
- The key to delivering robust software?
  - Designing an understandable, organized system.
  - Verifying that features of the system interact correctly.
  - **AKA: "taming the complexity"**
- **This course is about taming complexity.**
  - Exemplified primarily through software product lines.

Today's Goals

Introduce TDA 594/DIT 593 - Software Engineering Principles for Complex Systems

- AKA: What the heck is going on?
- Go over syllabus
- Clarify course expectations
- Assignments/grading
- Answer any questions
- Discuss software complexity
- Introduce the concept of **software product lines**

TDA/DIT 594 - SE Principles for Complex Systems



Complex?????????????

- **Variability**
  - The ability to change and customize software to deliver **variants** to new users.
  - Requires designing code **to be reused** and to **work with reused code**.
- **Changeability and Maintainability**
  - The ability to **add new features and options** while ensuring that **existing code still works**.

Software Product Lines

- Highly configurable families of systems.
- Built around common, modularized features.
  - Common set of core assets.
- Allows efficient development, customization.
- Examples:

android Cummins VOLVO BOEING

Why Software Product Lines?

- Designed **FOR** reuse of assets.
- Designed **TO** reuse assets.
- Successful SPLs are **highly configurable**, and **evolve easily** over time.
- Most modern systems intend to achieve at least one of these two tasks. **SPLs achieve both.**

Learning Outcomes

Knowledge and understanding

- Explain the challenges of engineering complex systems
- Explain industrial practice and examples of complex software systems engineering
- Explain processes and concepts for engineering complex and variant-rich software systems
- Explain business-, architecture-, process-, and organization-related aspects of engineering complex systems

Desired Course Outcomes

Skills and abilities

- Model a system from different perspectives
- Engineer a variant-rich system
- Analyze and re-engineer a complex system
- Use and reason about modularization techniques to increase cohesion and reduce coupling
- Use modern component or service frameworks

Desired Course Outcomes

Judgement and approach

- Analyze existing systems and discuss possible improvements or re-engineering potential
- Reason about software modularity concepts
- Recognize the situations in which certain of the taught principles are appropriate
- Read and analyze scientific literature

Teaching Team

- Examiner
  - Greg Gay (ggay@chalmers.se) **(Course Responsible)**
- Supervisors
  - Mukelabai Mukelabai (muka@chalmers.se)
  - Mazen Mohamad (mazenm@chalmers.se)
  - Wardah Mahmood (wardah@chalmers.se)
  - Shameer Kumar Pradhan (kumarsh@student.chalmers.se)

Student Representatives

- We want 5-6 Student Representatives
  - Interested?
    - E-mail ggay@chalmers.se if you want to volunteer.
  - We will pick random students if not enough volunteers.

Feedback! Questions!

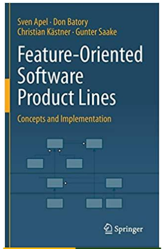
- Ask questions in Canvas discussion forum.
  - If private, contact teachers (about lectures), supervisors (assignments).
- If highly sensitive, contact me (Greg Gay)
- Course feedback should be sent to student representatives.
  - Send feedback early and often!

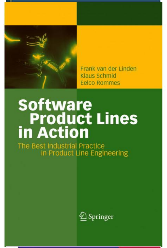
Course Content

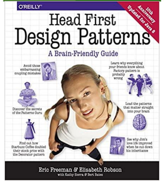
- https://chalmers.instructure.com/courses/16077
  - All lectures, files, reading goes here.
  - Pay attention to the schedule/announcements.
- https://greg4cr.github.io/courses/fall21tda594/
  - Backup of Canvas page/course materials.
  - Likely out of date, but may work if Canvas isn't working.

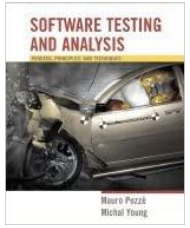
Course Literature

- Apel, S., Batory, D., Kästner, C., & Saake, G. **Feature-oriented software product lines.**
- Van der Linden, F. J., Schmid, K., & Rommes, E. **Software product lines in action: the best industrial practice in product line engineering.**
- Freeman E, Robson E., Bates B., & Sierra, K. **Head First Design Patterns.**
  - Free via Chalmers library.
- Pezze M., Young M. **Software Testing and Analysis: Process, Principles, and Techniques**
  - Free via https://ix.cs.uoregon.edu/~michal/book/free.php
- Additional literature will be provided via Canvas.


Sven Apel · Don Batory
Christian Kästner · Gunter Saake
Feature-Oriented Software Product Lines
Concepts and Implementation
Springer


Frank van der Linden
Klaus Schmid
Eelco Rommes
Software Product Lines in Action
The Best Industrial Practice in Product Line Engineering
Springer





Course Design

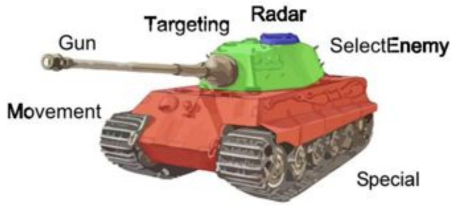
- Language of Instruction: English
- Lectures
  - Tuesday and Thursday, 13:15 - 15:00.
  - Attendance is highly recommended.
  - In-person and streamed (see Canvas for link).
- Supervision
  - Once per week, mandatory attendance.

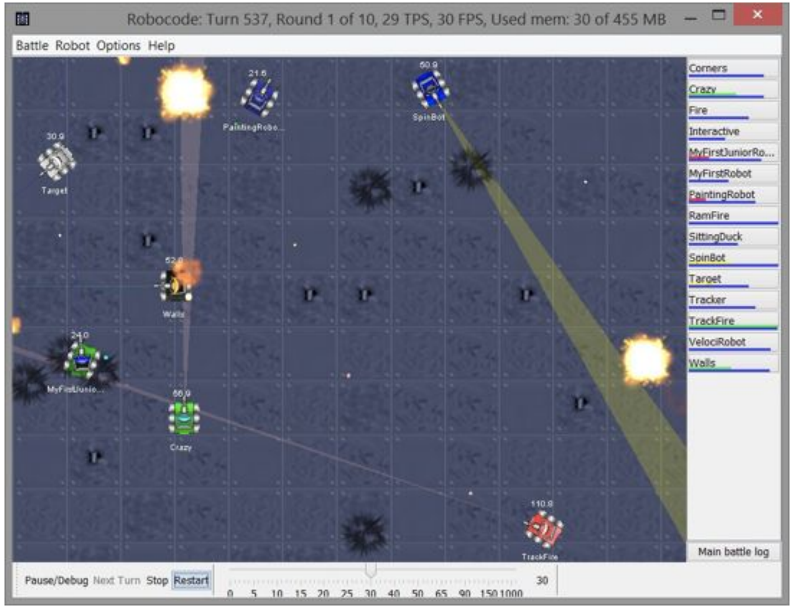
Changes Since Last Occasion

- Lectures topics have been revised.
    - Some topics dropped and replaced with more relevant material.
    - Revisions to existing slides for clarity.
- Assignments have been revised.
    - Streamlined group assignments.
    - Assignment content has been substantially revised.
    - Changes to format and grading of individual assignment.

Examination Form

- Group Project
  - 6.0 Credits, Scale: 3-5 and Fail (U)
  - 4-5 assignments
  - Groups of 6-7 students
  - Correspond to topics covered in the lectures
    - Case Studies, Domain Analysis, Modeling, Implementation, Testing of Software Product Lines.





For information about Robocode, see https://robocode.sourceforge.io/ and the RoboWiki (https://robowiki.net/wiki/Main_Page).

Forming Groups (Assignment 0)

- You may choose your groups.
  - **6-7 people**
- Submit one e-mail per group to ggay@chalmers.se
  - Names of all members. I will reply with a group number.
- Indicate in Canvas submission either team number or if you want to be assigned to a team.
- **Due Thursday, November 4, 11:59 PM**

Examination Form

- Written Assignment
  - 1.5 credits, Grading scale: 3-5 and Fail (U)
  - One assignment, at end of course.
  - Exam-like questions to assess critical understanding.
- All assignments must be submitted.
- Final grade in group project is average of assignment grades.
- Rubrics provided for each assignment.

Failing and Resubmission

- Late assignments:
  - -20%/loss of a grade level (5->4) (1 day), 40%/loss of two levels (5->3) (2 days), fail (3+ days).
- If final group average is failing, all group assignments must be resubmitted.
- If any assignment is failed, may resubmit two times.
  - Resubmissions due <= 1 month after course completion.
  - Must provide a changelog and explanation.

Other Policies

Integrity and Ethics:

Homework and programs you submit for this class must be entirely your own. If this is not absolutely clear, then contact me. Any other collaboration of any type on any assignment is not permitted. It is your responsibility to protect your work from unauthorized access. Violation = failing grade and reporting.

Classroom Climate:

Arrive on time, don’t talk during lecture, don’t use chat unless asking or answering questions. Disruptive students will be warned and dismissed.

Other Policies

Diversity

Students in this class are expected to work with all other students, regardless of gender, race, sexuality, religion, etc. Zero-tolerance policy for discrimination.

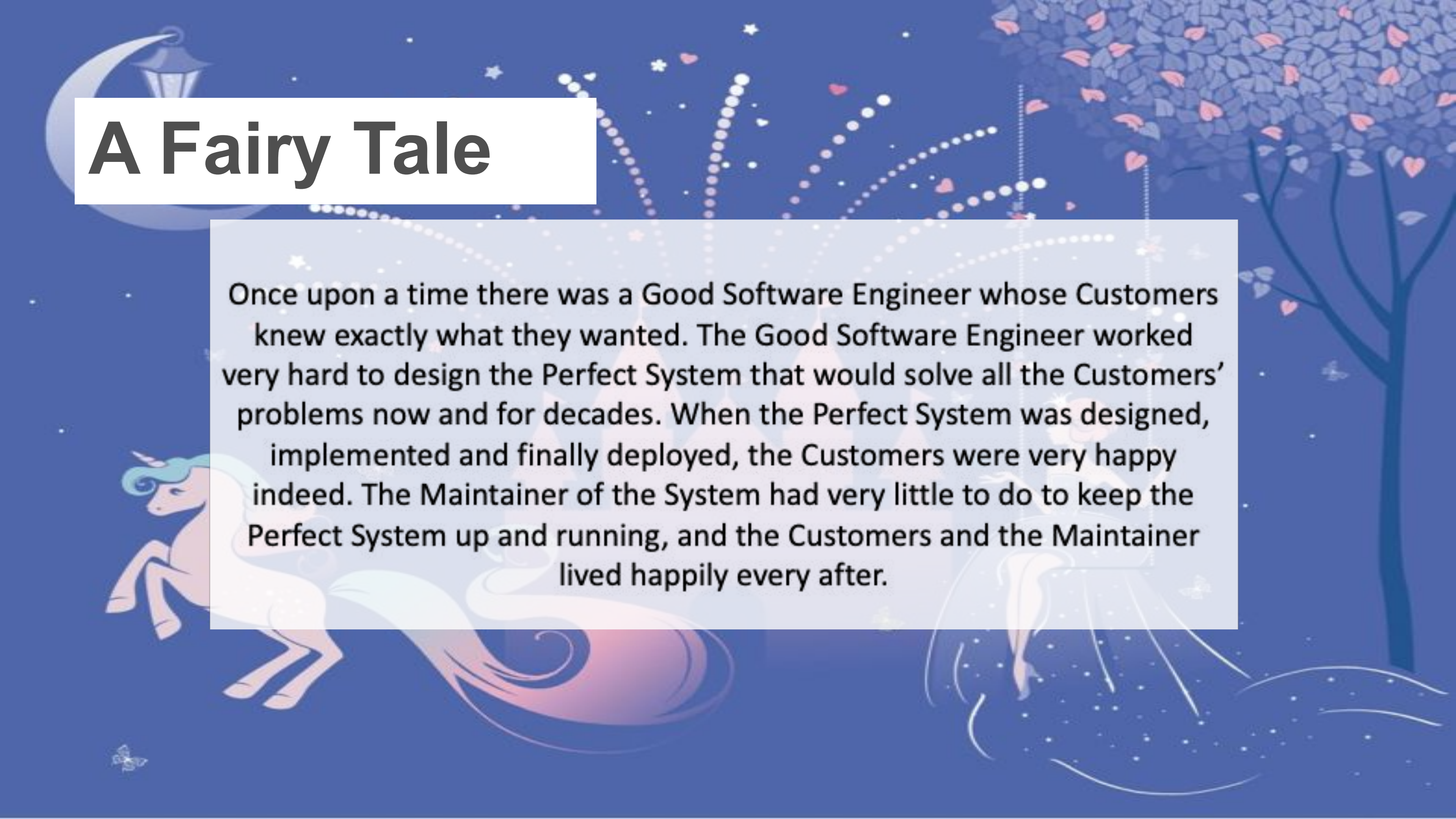
Special Needs

We will provide reasonable accommodations to students that have disabilities. Contact teaching team early to discuss individual needs.

Let’s take a break!

Complexity and Variability

A Fairy Tale

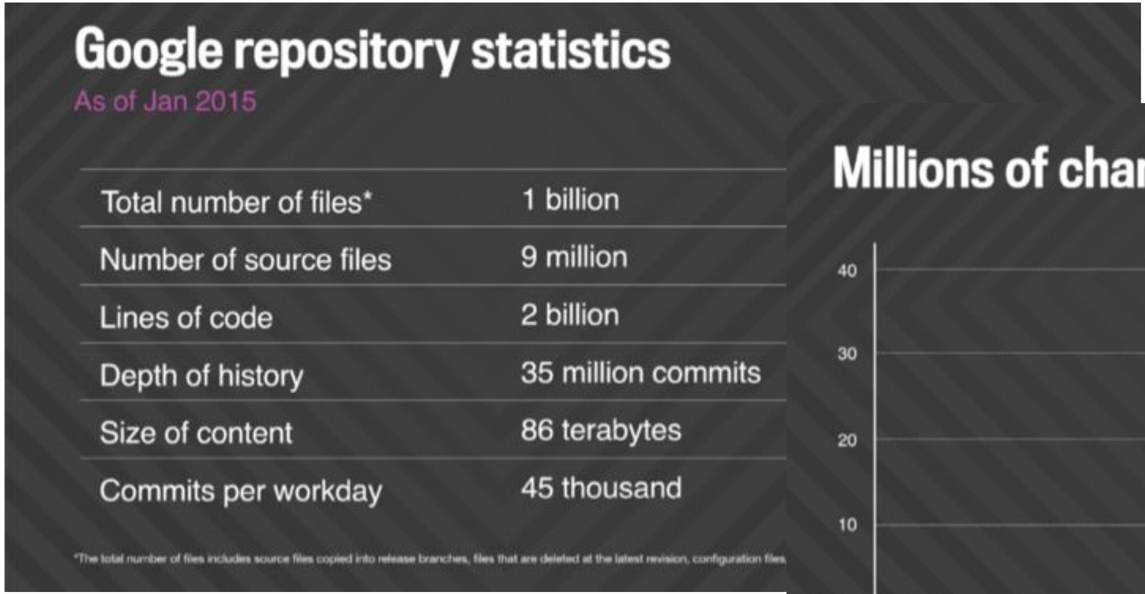


Once upon a time there was a Good Software Engineer whose Customers knew exactly what they wanted. The Good Software Engineer worked very hard to design the Perfect System that would solve all the Customers' problems now and for decades. When the Perfect System was designed, implemented and finally deployed, the Customers were very happy indeed. The Maintainer of the System had very little to do to keep the Perfect System up and running, and the Customers and the Maintainer lived happily every after.

Why Isn’t the Fairy Tale True?

- The Law of Continuing Change
  - A program used in a real-world environment must change, or become progressively less useful in that environment.
- The Law of Increasing Complexity
  - As a program evolves, it becomes more complex, and extra resources are needed to preserve and simplify its structure

Development at Google

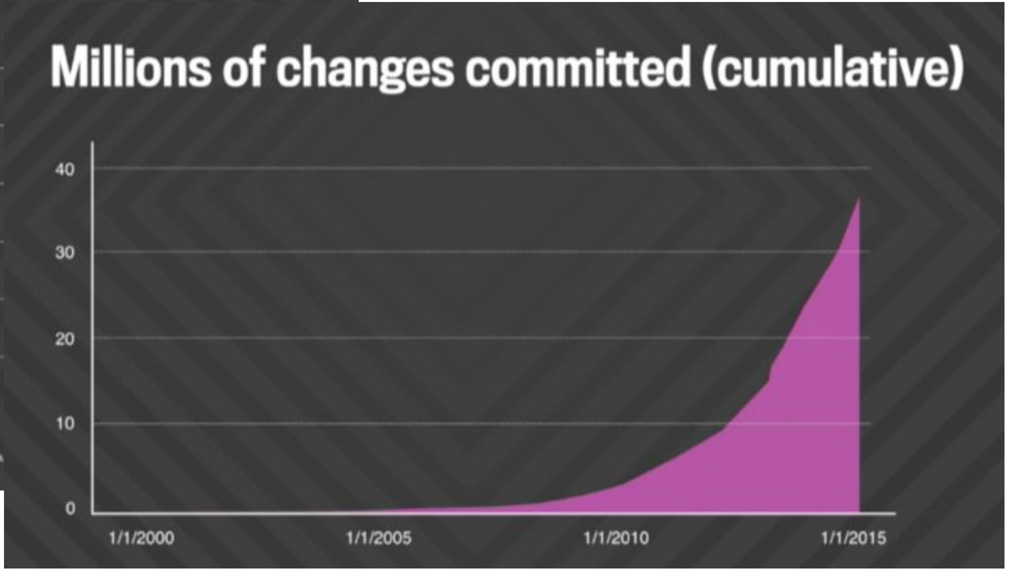


Google repository statistics

As of Jan 2015

Total number of files*	1 billion
Number of source files	9 million
Lines of code	2 billion
Depth of history	35 million commits
Size of content	86 terabytes
Commits per workday	45 thousand

*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files,



Development at Google

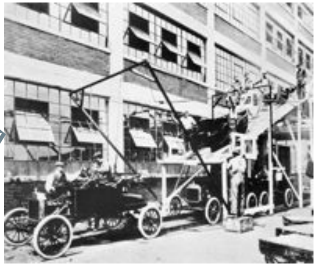
- >30,000 developers in 40+ offices
- 13,000+ projects under active development
- Single code tree with mixed language code
  - All builds from source
  - 30k commits per day (1 every 3 seconds)
  - 30+ sustained code changes per minute with 90+ peaks
  - 50% of code changes monthly
  - Requires execution of 150+ million test cases / day

Customization

Hand-Crafting

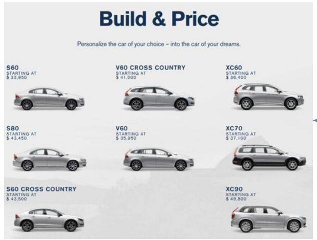
1
2
3
5
7
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
42

Replaceable Parts



Assembly Line

Automated Assembly Line

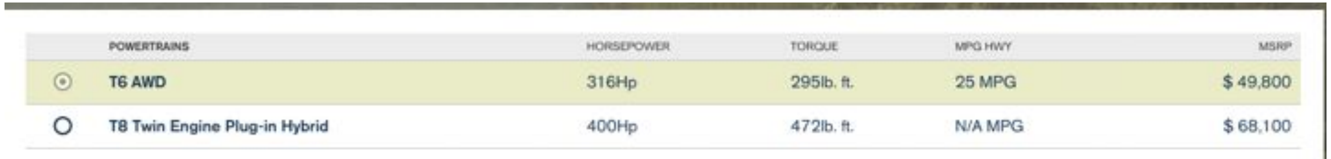

Build & Price
Personalize the car of your choice – into the car of your dreams.
S60 STARTING AT $ 33,950
V60 CROSS COUNTRY STARTING AT $ 41,000
XC60 STARTING AT $ 36,400
S80 STARTING AT $ 43,450
V60 STARTING AT $ 35,950
XC70 STARTING AT $ 37,100
S60 CROSS COUNTRY STARTING AT $ 43,500
XC90 STARTING AT $ 49,800

Product Lines

Customization ⟷ **Standardization**

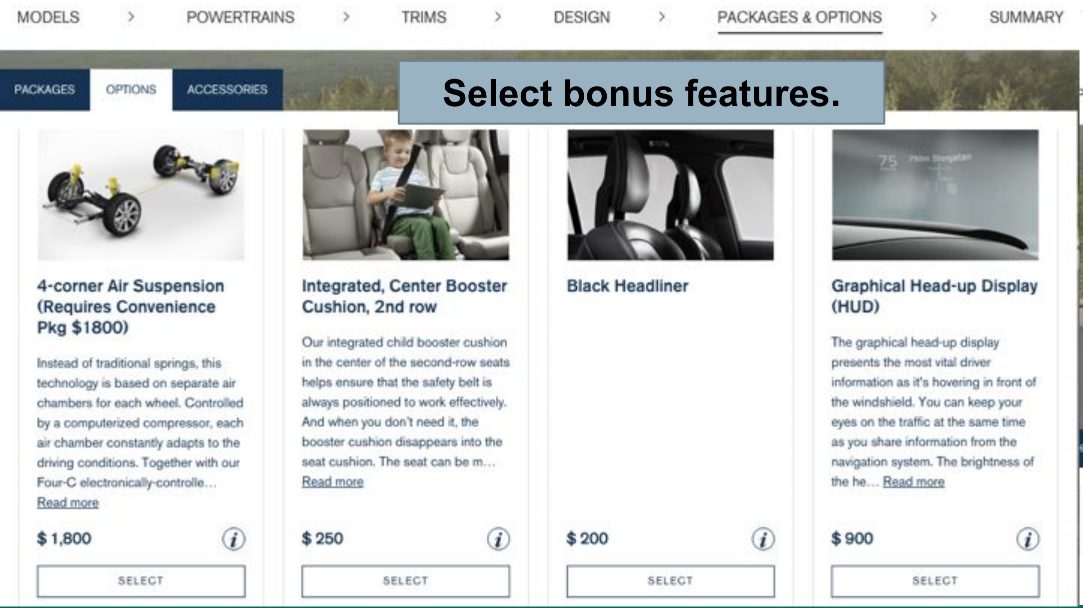
Mass Customization

Volvo XC90



	POWERTRAINS	HORSEPOWER	TORQUE	MPG HWY	MSRP
◉	T6 AWD	316Hp	295lb. ft.	25 MPG	$ 49,800
○	T8 Twin Engine Plug-in Hybrid	400Hp	472lb. ft.	N/A MPG	$ 68,100

Choose your engine...



Select bonus features.

4-corner Air Suspension (Requires Convenience Pkg $1800)

Instead of traditional springs, this technology is based on separate air chambers for each wheel. Controlled by a computerized compressor, each air chamber constantly adapts to the driving conditions. Together with our Four-C electronically-controlle… Read more

$ 1,800

SELECT

Integrated, Center Booster Cushion, 2nd row

Our integrated child booster cushion in the center of the second-row seats helps ensure that the safety belt is always positioned to work effectively. And when you don't need it, the booster cushion disappears into the seat cushion. The seat can be m… Read more

$ 250

SELECT

Black Headliner

$ 200

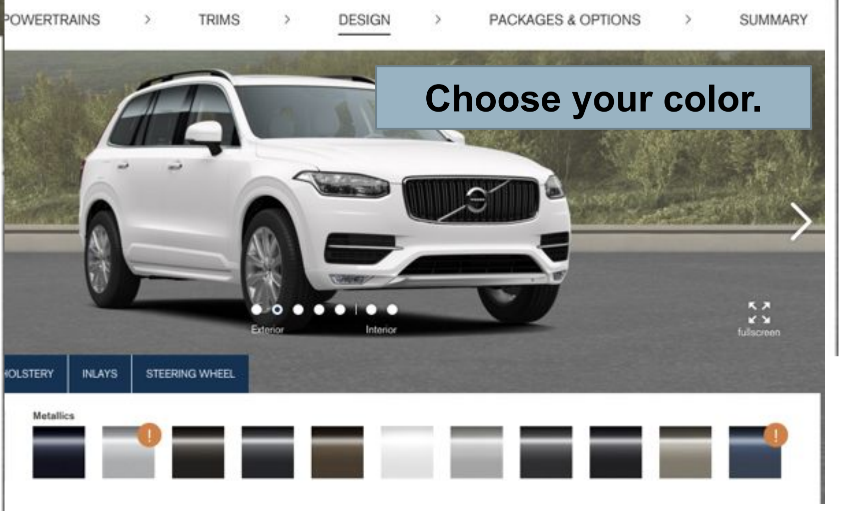
SELECT

Graphical Head-up Display (HUD)

The graphical head-up display presents the most vital driver information as it's hovering in front of the windshield. You can keep your eyes on the traffic at the same time as you share information from the navigation system. The brightness of the he… Read more

$ 900

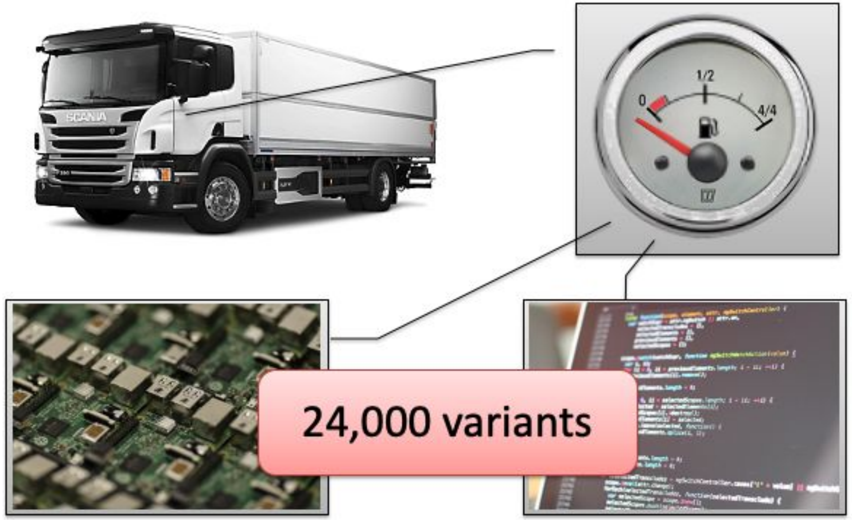
SELECT



Choose your color.

BMW X3, Scania Fuel Indicator

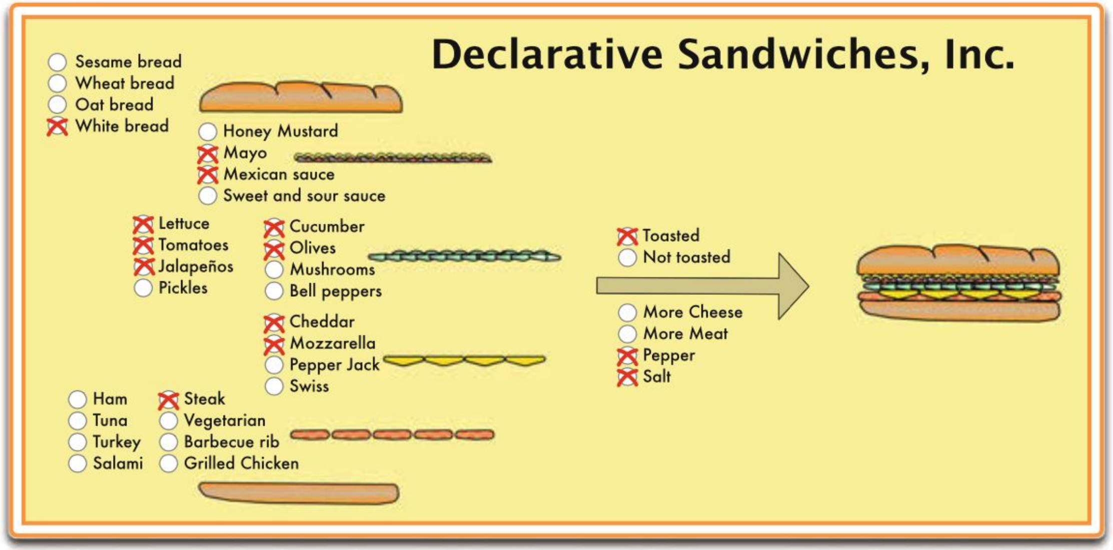
Roofliner:	Doors:	Rear axles:
90,000	**3,000**	**324**
variants	variants	variants



Number of Variants Explodes

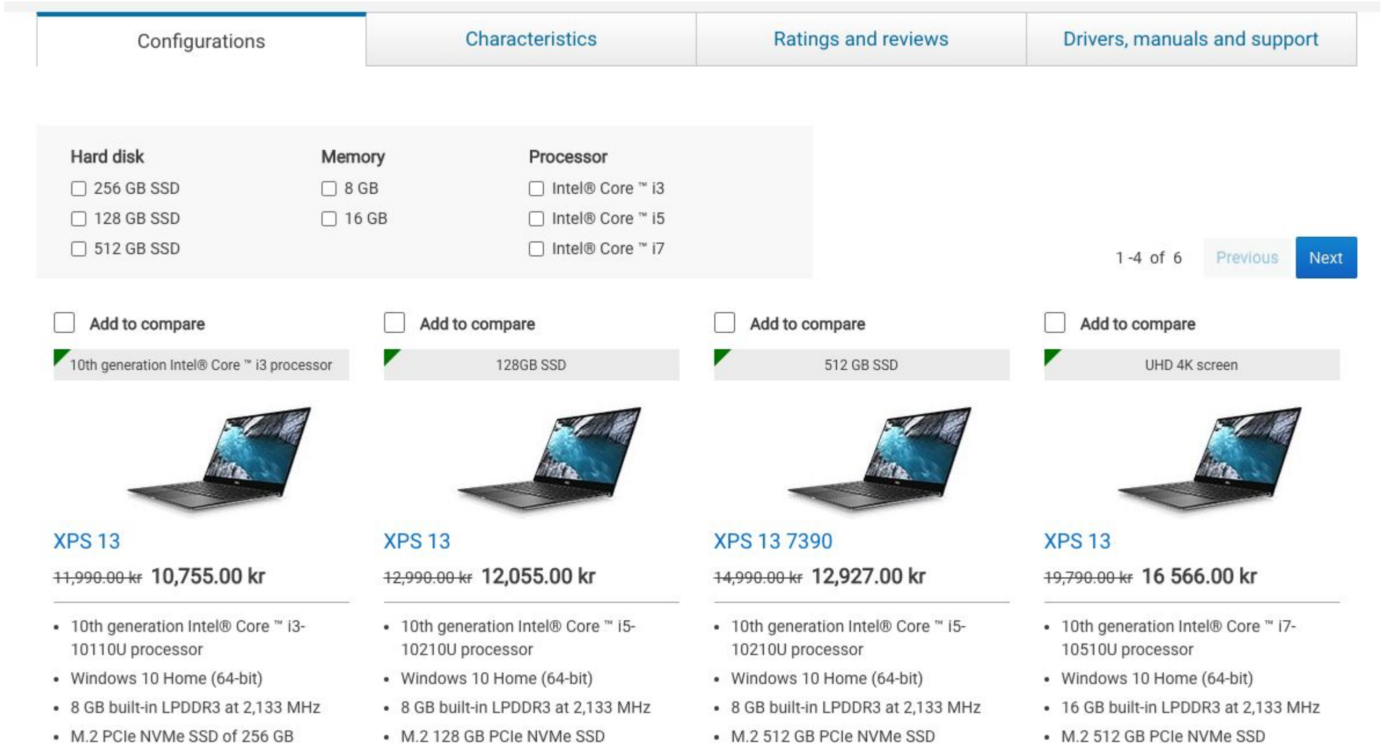
- 30 years ago: **Mass Production**
  - Few extras (alternative cassette player or roof rack)
  - Single variant (Audi 80, 1.3l, 55hp) 40% of all sales
- Now: **Mass Customization**
  - 10^{20} configurations at Audi, 10^{32} configurations at BMW
    - 100 different undercarriages, 50 steering wheels
  - **Identical configurations are rarely produced**
    - **This is OK because of shared assets.**

The Sandwich Shop



- Sandwiches are build to match customer requests.
- Not arbitrary. Customer chooses from a set of successive choices.
  - Meats, toppings, sauces.
  - **Shared assets.**

Customized Computers


Configurations | Characteristics | Ratings and reviews | Drivers, manuals and support

Hard disk
- ☐ 256 GB SSD
- ☐ 128 GB SSD
- ☐ 512 GB SSD

Memory
- ☐ 8 GB
- ☐ 16 GB

Processor
- ☐ Intel® Core ™ i3
- ☐ Intel® Core ™ i5
- ☐ Intel® Core ™ i7

1 -4 of 6 Previous Next

☐ Add to compare

10th generation Intel® Core ™ i3 processor

XPS 13

~~11,990.00 kr~~ 10,755.00 kr

- 10th generation Intel® Core ™ i3-10110U processor
- Windows 10 Home (64-bit)
- 8 GB built-in LPDDR3 at 2,133 MHz
- M.2 PCIe NVMe SSD of 256 GB

☐ Add to compare

128GB SSD

XPS 13

~~12,990.00 kr~~ 12,055.00 kr

- 10th generation Intel® Core ™ i5-10210U processor
- Windows 10 Home (64-bit)
- 8 GB built-in LPDDR3 at 2,133 MHz
- M.2 128 GB PCIe NVMe SSD

☐ Add to compare

512 GB SSD

XPS 13 7390

~~14,990.00 kr~~ 12,927.00 kr

- 10th generation Intel® Core ™ i5-10210U processor
- Windows 10 Home (64-bit)
- 8 GB built-in LPDDR3 at 2,133 MHz
- M.2 512 GB PCIe NVMe SSD

☐ Add to compare

UHD 4K screen

XPS 13

~~19,790.00 kr~~ 16 566.00 kr

- 10th generation Intel® Core ™ i7-10510U processor
- Windows 10 Home (64-bit)
- 16 GB built-in LPDDR3 at 2,133 MHz
- M.2 512 GB PCIe NVMe SSD

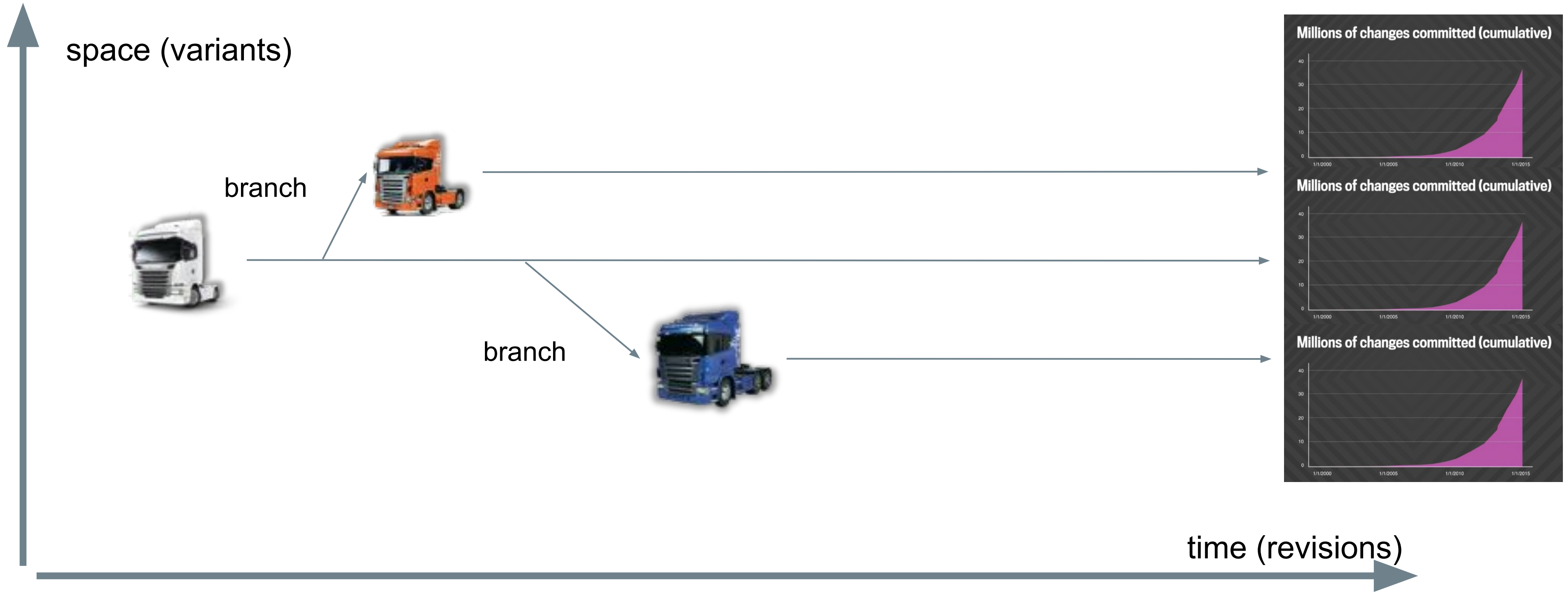
Standardized Software

- Industry agreed on standard platforms
  - Ex: Excel instead of your own spreadsheet program.
- Tries to satisfy the needs of most customers.
  - For most customers, many features are not needed.
  - Can lead to software that is hard to use, slow, buggy.
  - May still be missing features, not support hardware, ...

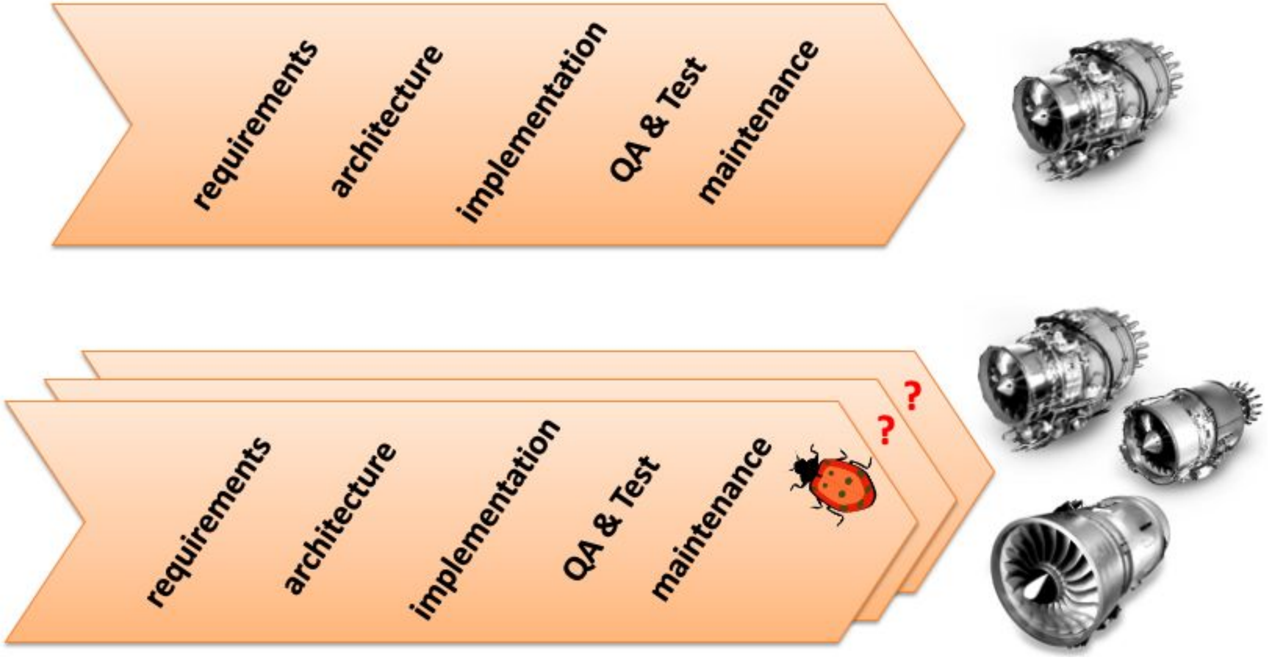
Customized Software

- Personalization
    - individual requirements, look & feel, special algorithms
- Hardware
    - robots (sensors and actuators), computers (hardware devices)
- Resource restrictions
    - energy consumption, performance, memory demand
- Software and product quality
    - usability
    - maintenance/verification/validation effort grows with functionality

Complexity Grows in Time and Space


space (variants)
branch
branch
Millions of changes committed (cumulative)
Millions of changes committed (cumulative)
Millions of changes committed (cumulative)
40
30
20
10
0
1/1/2000
1/1/2005
1/1/2010
1/1/2015
time (revisions)

Variability Adds Complexity



Variability in Software Ecosystems

	eCos	Linux Kernal	Eclipse	Debian	Android
What:	embedded OS,	OS kernal,	IDE,	desktop/server OS,	mobile OS
Customization:	config options,	config options,	add-ons,	packages,	apps (event handling)

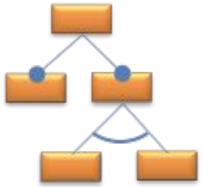
Variability in Software Ecosystems

	eCos	Linux Kernal	Eclipse	Debian	Android
Options	2.8K	13K	> 7K	28K	> 3.3M

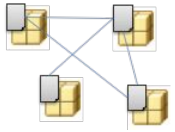
Controlling Variability

- **Variability mechanisms** introduced controlled points where customization can occur.
  - called **variation points**
- Variation points determine the concrete software variants that can occur.
  - In a closed system, we know how many variants are possible (Linux kernal)
  - In an open system, we cannot know all possible variants
    - (Android - ex: different apps can be bound to variation points)

Variability Mechanisms



- Linux and eCos use **feature models**
  - Hierarchical menus of config options
  - Centralized list of options, constraints limit combinations.



- Debian and Eclipse use **manifest-based packages.**
  - Structured metadata about the options.
  - Version info, dependencies, categorization.
  - Central repositories

- Android uses **service-oriented apps.**
  - Provide a common interface and constraints on capabilities
  - Hook to concrete apps at run-time.

Variability Management

- Friction between **managing** variability and **encouraging** it.
  - Management allows control of scope, limitation of number of variants.
  - However, encouragement unleashes innovation and encourages competition.
- Many options we will cover in class will help manage and control variability.

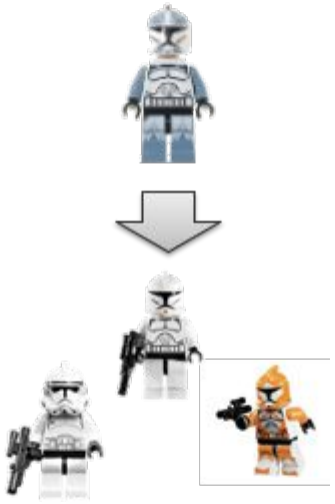
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Software Product Lines

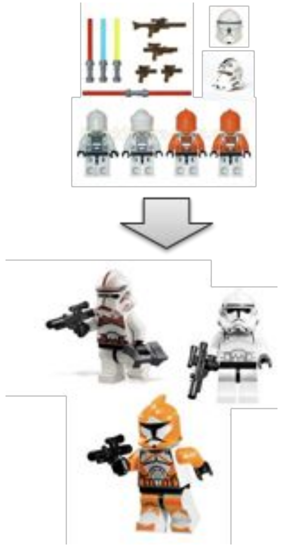
Software Customization

Build Independently

Clone & Own



Shared Assets



Software Product Lines

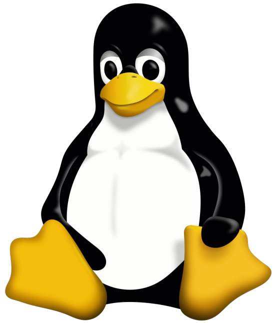
- Build variants from reusable **shared assets**.
  - Customers select from configuration options.
  - Assets = code components, interfaces, other resources.
- Enables customization, while retaining benefits of mass production.
  - Avoids explosion in "space" as we manage the portfolio of assets instead of each individual variant.

Feature-Oriented Approach

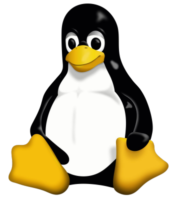
- Features distinguish products from a product line.
    - Editor Version A has spell-check. Version B does not.
    - E-mail client A supports IMAP and POP3. Client B supports only POP3.
- Product line artifacts and process structured around features and feature interactions.
    - Discuss control, implementation, verification of features.
    - Connects requirements to customizations in code.

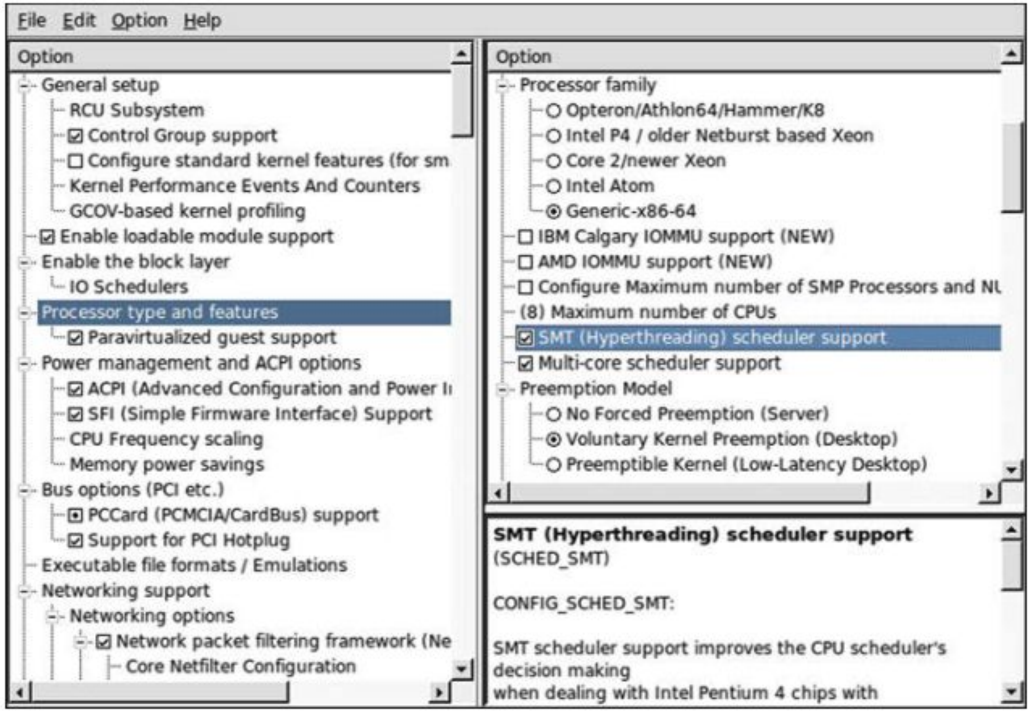
Example: Linux Kernal

- > 8M Lines of Code
- Supports > 22 hardware architectures
- > 13,000 configuration options



Ex: Linux Configuration Tools





```
File  Edit  Option  Help

Option
⊟ General setup
    RCU Subsystem
    ☑ Control Group support
    ☐ Configure standard kernel features (for sm
    Kernel Performance Events And Counters
    GCOV-based kernel profiling
  ☑ Enable loadable module support
⊟ Enable the block layer
    IO Schedulers
⊟ Processor type and features
    ☑ Paravirtualized guest support
⊟ Power management and ACPI options
    ☑ ACPI (Advanced Configuration and Power I
    ☑ SFI (Simple Firmware Interface) Support
    CPU Frequency scaling
    Memory power savings
⊟ Bus options (PCI etc.)
    ⊡ PCCard (PCMCIA/CardBus) support
    ☑ Support for PCI Hotplug
  Executable file formats / Emulations
⊟ Networking support
  ⊟ Networking options
    ⊟ ☑ Network packet filtering framework (Ne
        Core Netfilter Configuration
```

```
Option
⊟ Processor family
    ○ Opteron/Athlon64/Hammer/K8
    ○ Intel P4 / older Netburst based Xeon
    ○ Core 2/newer Xeon
    ○ Intel Atom
    ◉ Generic-x86-64
  ☐ IBM Calgary IOMMU support (NEW)
  ☐ AMD IOMMU support (NEW)
  ☐ Configure Maximum number of SMP Processors and NU
  (8) Maximum number of CPUs
  ☑ SMT (Hyperthreading) scheduler support
  ☑ Multi-core scheduler support
⊟ Preemption Model
    ○ No Forced Preemption (Server)
    ◉ Voluntary Kernel Preemption (Desktop)
    ○ Preemptible Kernel (Low-Latency Desktop)
```

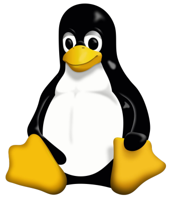
```
SMT (Hyperthreading) scheduler support
(SCHED_SMT)

CONFIG_SCHED_SMT:

SMT scheduler support improves the CPU scheduler's
decision making
when dealing with Intel Pentium 4 chips with
```

- Kconfig used to tailor Linux kernal for custom use.
- Compiles a custom Linux kernal with only the selected options.
- Omits unnecessary or incompatible features.

Configuration Options



- Linux kernal customization managed through a **feature model**.
  - Configuration options are features.
  - Constraints are used to prevent illegal combinations.
- #IFDEF in C/C++ allows conditional compilation.
  - If we select this option, the contained code will be compiled into the custom build. If not, it is omitted entirely.
  - **Unique build tailored to choices.**

(3.9: arch/ia64/kernel/irq_ia64.c)

```
...
void __init
init_IRQ (void)
{
#ifdef CONFIG_ACPI
        acpi_boot_init();
#endif
        ia64_register_ipi();
        register_percpu_irq(...);
...
}
```

Constraints on Features

```
#ifdef ASH
  #ifdef NOMMU
  #error “... ASH will not run on NOMMU”
  #endif

 #ifdef EDITING
 void init() {
   initEditing();
   int maxlength = 1 *
     #ifdef EDITING_MAX_LEN
       EDITING_MAX_LEN;
     #endif
 }
 #endif
#endif
```

Preprocessor error if:
(ASH ∧ NOMMU)

Parser error if:
(ASH ∧ EDITING ∧ ¬ EDITING_MAX_LENGTH)

Feature models prevent these errors via feature constraints.

Benefits of Product Lines

- Products tailored to specific customers.
- Reduced costs, fast start-up, reduced complexity through asset reuse.
  - Slower initial effort, but it pays off quickly.
- Improved quality from testing assets in isolation.
  - Frequently used assets will be heavily tested over time.
- Can quickly produce new variants or change variants to respond to market conditions.

Success Stories

- Boeing
- Bosch Group
- Cummins, Inc.
- Ericsson
- General Dynamics
- General Motors
- Hewlett Packard
- Lockheed Martin
- Lucent
- NASA
- Nokia
- Philips
- Siemens
- Volvo
- … (more success stories at https://splc.net/fame.html)

Hewlett Packard Printer Firmware

- > 2000 features
- Hundreds of printer models.
- Production costs reduced by 75%.
- Development time reduced by 33%.
- Reported defects reduced by 96%.

We Have Learned

- Modern software is complex!
  - Variability: The ability to change and customize software to deliver variants to new users.
  - Changeability and Maintainability: The ability to add new features while ensuring that existing code still works.
- Software must evolve to remain useful.
  - Evolution introduces additional complexity.
- Software Product Lines are designed to enable and take advantage of reuse.

Next Time

- Introduction to Domain and Application Engineering

- Plan your team selection.
  - The earlier, the better!
    - **Due Thursday, November 4, 11:59 PM.**
  - 6-7 people, e-mail names to ggay@chalmers.se
  - Submit group number or desire to be assigned to a team on Canvas.

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY