

Konfigurierbare Systemsoftware (KSS)

VL 5 – Variability Management in the Large: The VAMOS Approach

Daniel Lohmann

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

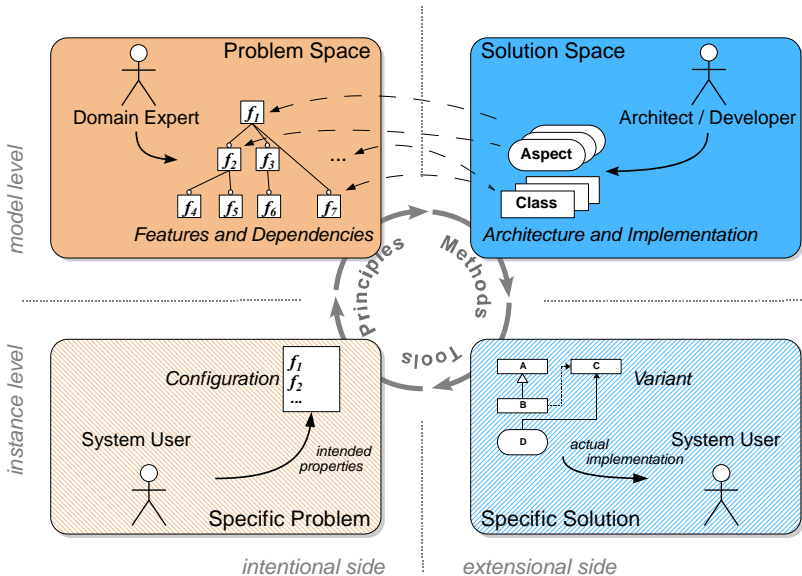
Friedrich-Alexander-Universität
Erlangen-Nürnberg

SS 12 – 2012-06-13

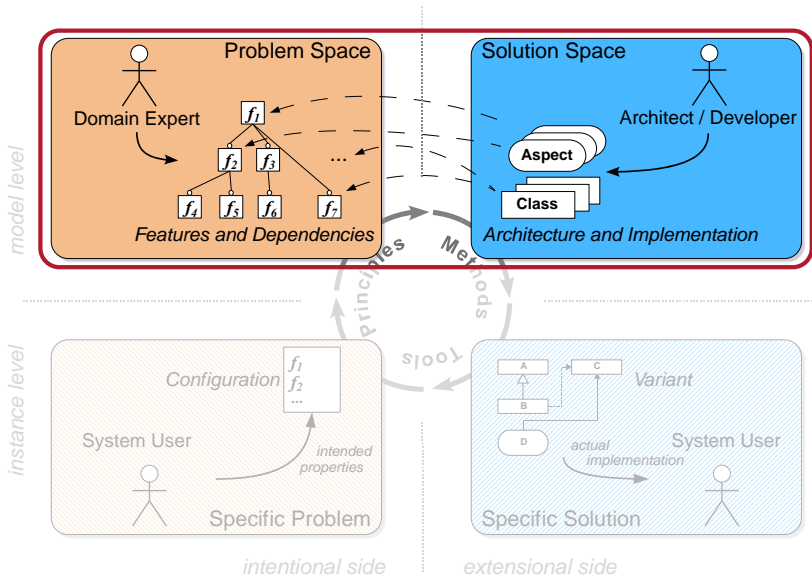
http://www4.informatik.uni-erlangen.de/Lehre/SS12/V_KSS



About this Lecture



About this Lecture

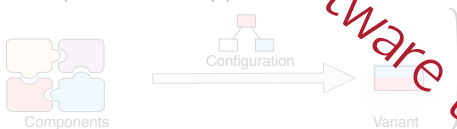


Decompositional Approaches



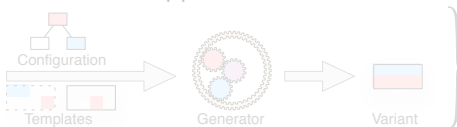
- Text-based filtering (untyped)
- Preprocessors

Compositional Approaches



- Language-based composition mechanisms (typed)
- OOP, AOP, Templates

Generative Approaches



- Metamodel-based generation of components (typed)
- MDD, C++ TMS, generators

Real-world software uses them all!



Agenda

- 5.1 Motivation
- 5.2 Variability in Linux
- 5.3 Configuration Consistency
- 5.4 Configuration Coverage
- 5.5 Summary
- 5.6 References



33 optional, independent features



one individual variant
for each human being

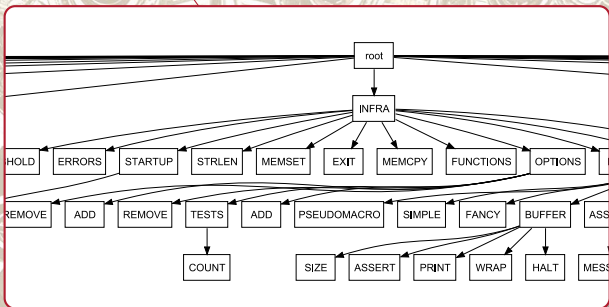
optional, independent
320 features

more variants than
atoms in the universe!

Typical Configurable Operating Systems...



1,250 features



Typical Configurable Operating Systems...

ecos

1,250 features

Challenges:



VAMOS*

- How to maintain this?
- How to test this?
- Why so many features anyway?

* [VA](#)riability [M](#)anagement in [O](#)perating [S](#)ystems



12,000 features

5.1 Motivation

5.2 Variability in Linux

Variability Implementation in Linux

Challenges

5.3 Configuration Consistency

5.4 Configuration Coverage

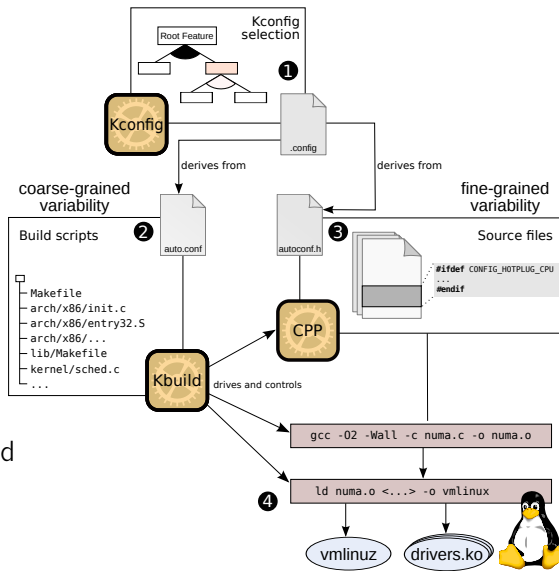
5.5 Summary

5.6 References



The Linux Configuration and Generation Process

- 1 Configuration with an KCONFIG frontend
- 2 Compilation of a subset of files
- 3 Selection of a subset of CPP Blocks
- 4 Linking of the kernel and loadable kernel modules



Dominancy and Hierarchy of Variability

KCONFIG controlled Variability

l_0 : Feature Modelling 12,000 features

l_1 : Coarse-grained: KBUILD 31,000 source files

l_2 : Fine-grained: CPP 89,000 #ifdef blocks

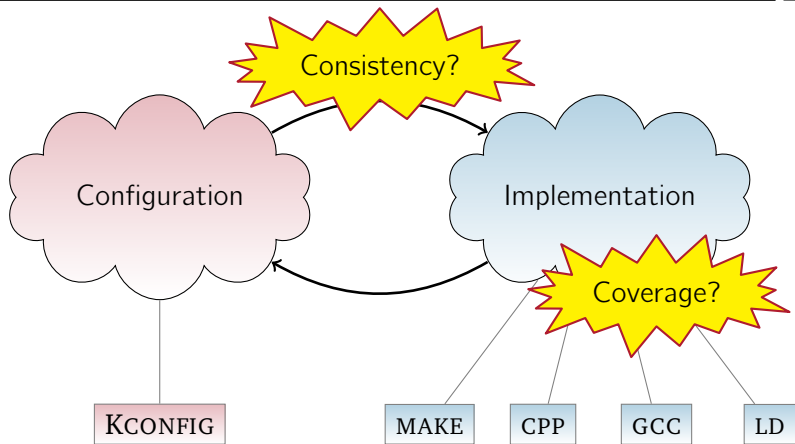
l_3 : Language-level: GCC → if(CONFIG_SMP) ...

l_4 : Link-time: LD → branches in linker scripts

l_5 : Run-time: INSMOD, MODPROBE, ...



Challenges with Implemented Variability



- **Central declaration** of configurability: KCONFIG
- **Distributed implementation** of configurability: MAKE, CPP, GCC, LD



5.1 Motivation

5.2 Variability in Linux

5.3 Configuration Consistency

Problem Analysis

Solution Approach

Results

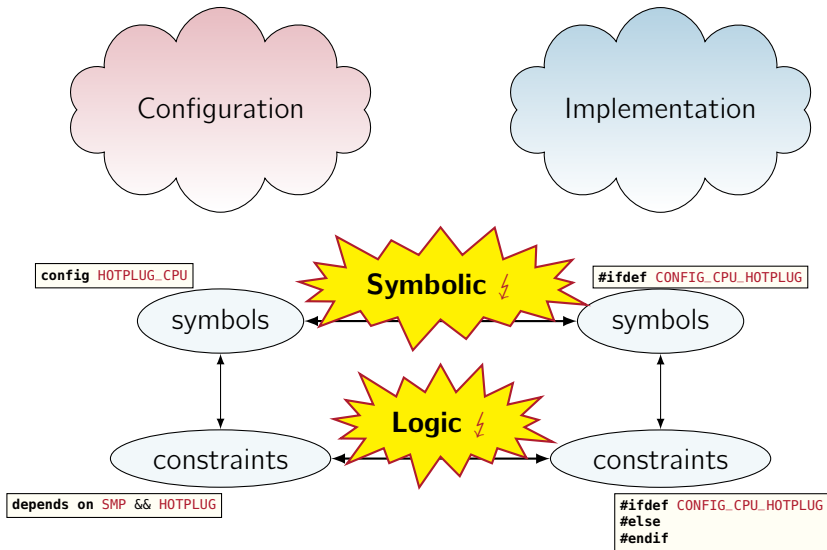
5.4 Configuration Coverage

5.5 Summary

5.6 References




Problem Analysis: Configuration Consistency



```
config HOTPLUG_CPU
    bool "Support for hot-pluggable CPUs"
    depends on SMP && HOTPLUG
    ---help---
```

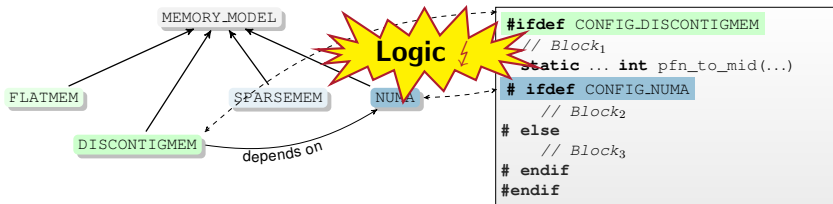
```
static int
hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu)
{
    // [...]
    switch (action) {
    case CPU_UP_PREPARE:
    case CPU_UP_PREPARE_FROZEN:
    // [...]
#ifdef CONFIG_CPU_HOTPLUG
    case CPU_UP_CANCELED:
    case CPU_UP_CANCELED_FROZEN:

    case CPU_DEAD:
    case CPU_DEAD_FROZEN:
        free_cpumask_var(cfd->cpumask);
        break;
#endif
    };
    return NOTIFY_OK;
}
```



Result:
Fix for a
critical bug





- Feature DISCONTIGMEM **implies** feature NUMA
 - Inner blocks are not actually configuration-dependent
 - *Block₂* is **always** selected ↪ **undead**
 - *Block₃* is **never** selected ↪ **dead**
- } **configurability defects**

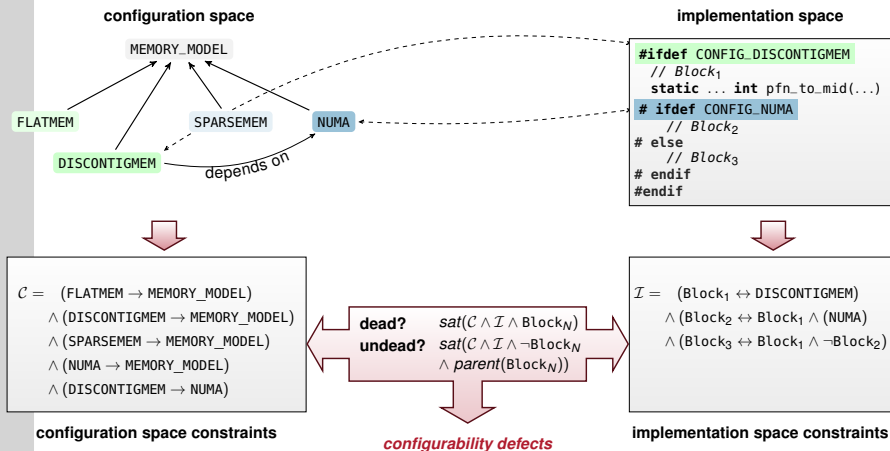
~ Linux contains **superfluous** #ifdef Blocks!

Result:
Code cleanup



Solution Approach: Consistency Validation

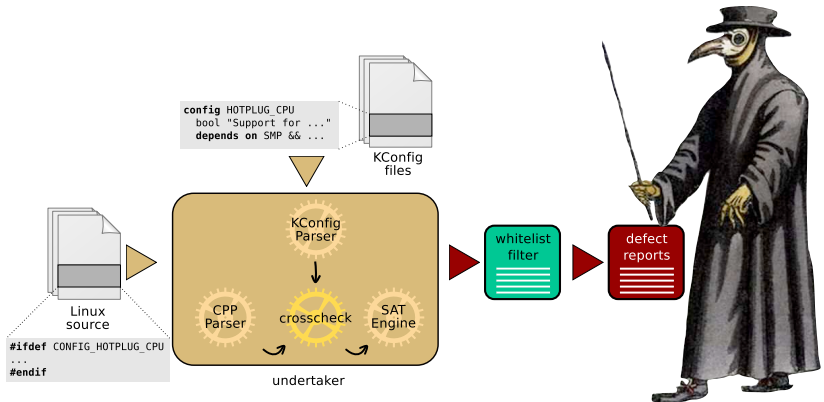
Problem and solution space are analyzed for configuration points:



⇒ and transformed into **propositional formulas**



Job: Find (and eventually bury) **dead #ifdef-code!**

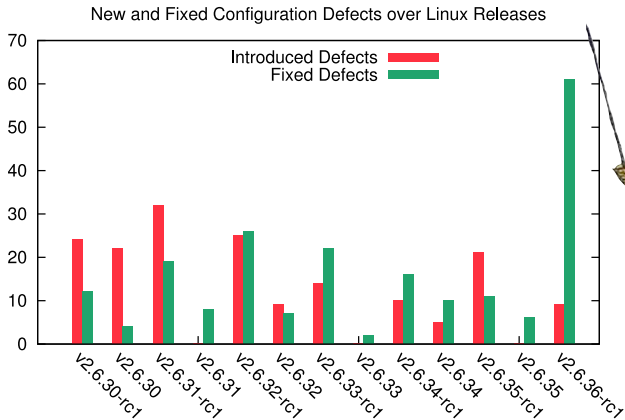


Job: Find (and eventually bury) **dead #ifdef-code!**

- We have found **1776** configurability defects in Linux v2.6.35
- Submitted **123** patches for **364** defects
- **20** are confirmed **new bugs** (affecting binary code)
- Cleaned up **5129** lines of **craft code**



Job: Find (and eventually bury) **dead #ifdef-code!**



How good is this, *really?*



5.1 Motivation

5.2 Variability in Linux

5.3 Configuration Consistency

5.4 Configuration Coverage

Where Have All the Features Gone?

Results

Extracting Variability from KBUILD

Improvements

5.5 Summary

5.6 References



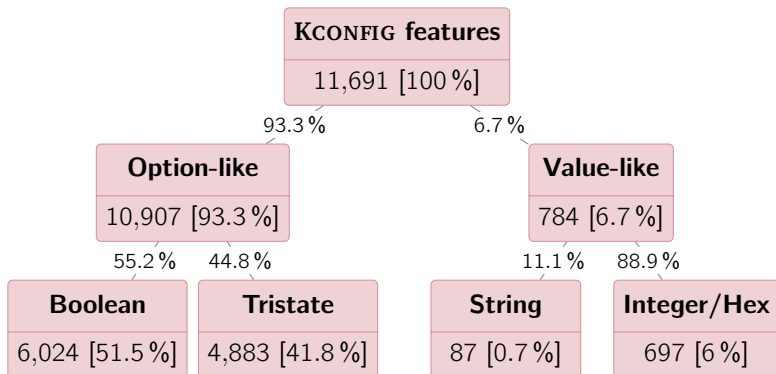
Common Beliefs About Variability in Linux

- 1 Most variability is expressed by boolean (or tristate) switches.
- 2 arch-x86 is the largest and allyesconfig selects most features.
- 3 Variability is mostly implemented with the CPP.
- 4 The Linux *kernel* is highly configurable.



Linux v3.1: Feature Distribution by Type

- 1 Most variability is expressed by boolean (or tristate) switches

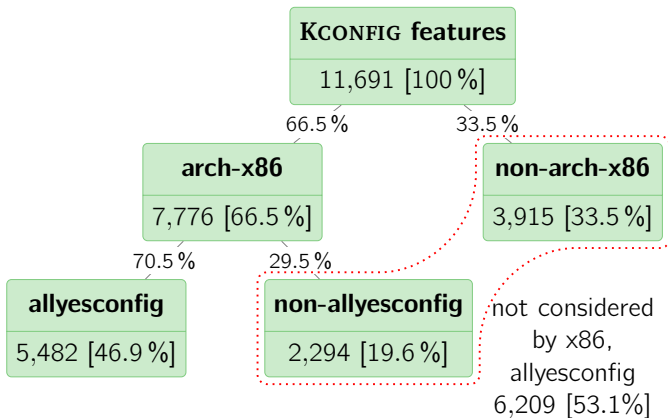


⇒ Almost all features in Linux are **option-like**



Linux v3.1: Coverage of arch-x86 / allyesconfig

- ② arch-x86 is the largest and allyesconfig selects most features

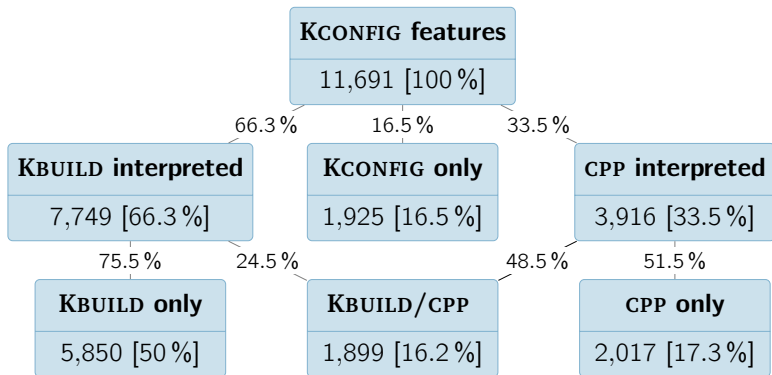


⇒ arch-x86/allyesconfig is **not nearly** a full configuration



Linux v3.1: Distribution by Granularity

- ③ Variability is mostly implemented with the CPP

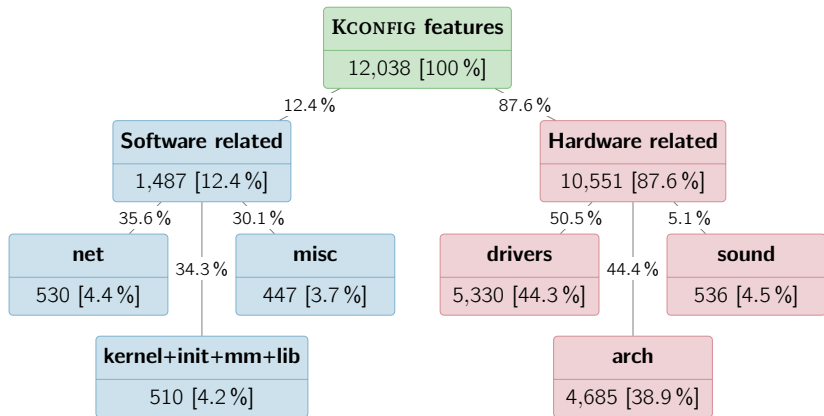


⇒ KBUILD implements **more than two thirds** of all variation points



Linux v3.2: Distribution by HW/SW

- ④ The Linux *kernel* is highly configurable

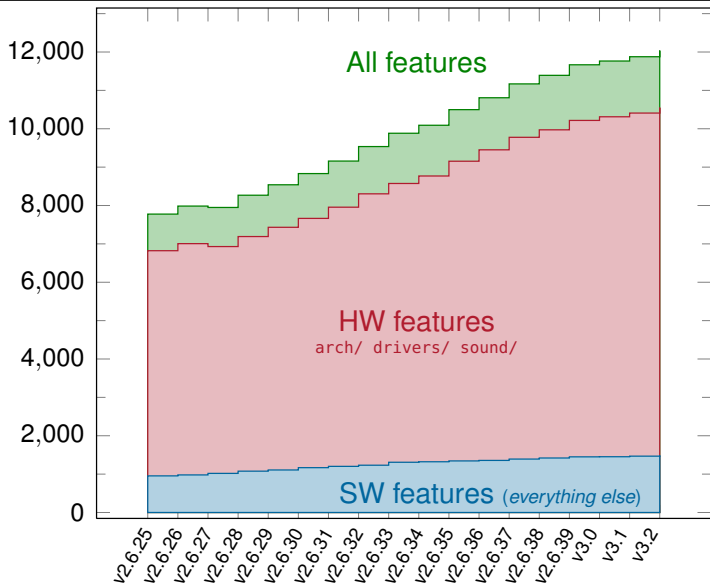


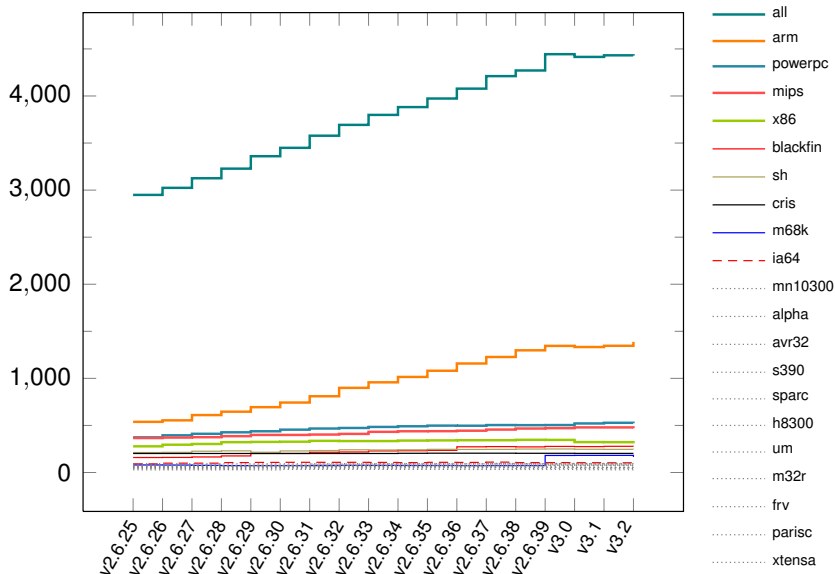
⇒ Software features account for
only twelve percent of all variation points



Linux Feature Growth over Time

(#Features, 2007–2012)





Results: Where Have all the Features Gone?

- ① Most variability is expressed by boolean (or tristate) switches ✓
 - more than 93 percent of all features are option-like
 - ↪ it is acceptable for tools to ignore value-type features
- ② arch-x86 is the largest and allyesconfig selects most features ✗
 - more than 53 percent are not covered by this configuration
 - ↪ other parts of Linux are probably less tested and error-prone!
- ③ Variability is mostly implemented with the CPP ✗
 - more than 66 percent of all features are handled by the build system, only 17 percent are handled by C++ only
 - ↪ variability extraction from `KBUILD` is necessary
- ④ The Linux *kernel* is highly configurable ✗
 - only 12 percent of all features configure software only
 - variability is mostly induced by advances in hardware
 - ↪ complexity will increase further



Challenges: Variability Extraction from the Build System

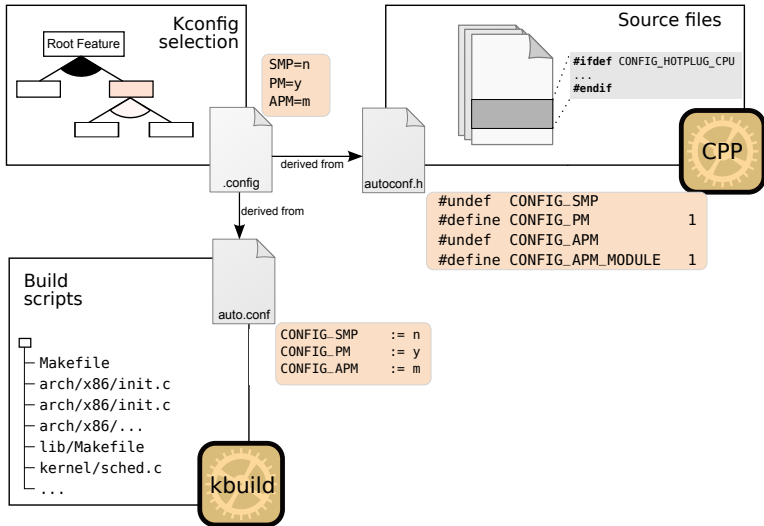
- Variability extraction \mapsto which file is selected by which feature?
- Usual approach for variability extraction [5, 7] (KCONFIG, CPP, ...):



- **Parsing does not work** well for MAKE-languages
 - declarative and Turing-complete languages
 - special features, like `shell`, `foreach`, `eval`, `addprefix`, ...
- Linux's `KBUILD` is built on top of (GNU) `MAKE`
 - nevertheless, researchers have tried parsing to extract variability
 - `KBUILDMINER` by Berger, She, Czarnecki, et al. [1]
 - `Nadi` parser by Nadi and Holt [4]
 - resulting tools are too **brittle at best**
 - work for a (few) Linux version(s) only
 - each usage of a special feature requires manual tailoring



Linux Build Process Revisited



Basic idea: Systematic probing and inferring of implications

SPLC '12: Dietrich, et al. [2]

- *Dancing Makefiles*
- Identification of KCONFIG references
- Recursion into subdirectory while considering constraints

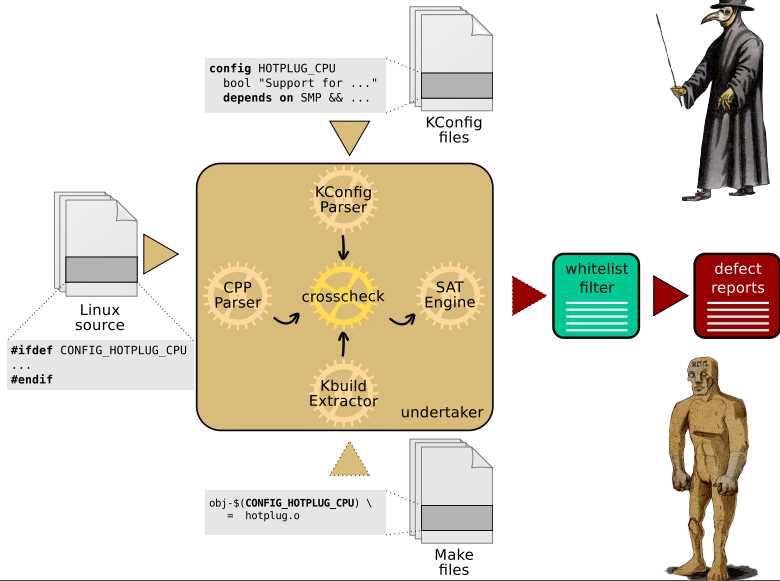
```
obj-y += fork.o
obj-$(CONFIG_SMP) += spinlock.o
obj-$(CONFIG_APM) += apm.o
```

```
obj-$(CONFIG_PM) += power/
```

- Robust with respect to architecture and version
- ⇒ **no adaptations** on or for KBUILD!

Kernelversion	found inferences	
v2.6.25	6,274	(93.7%)
v2.6.28.6	7,032	(93.6%)
v2.6.33.3	9,079	(94.9%)
v2.6.37	10,145	(95.1%)
v3.2	11,050	(95.4%)





Configuration defects in Linux v3.2:

Without KBUILD constraints

Code defects	1835
Referential defects	415
Logical defects	83
Sum:	Σ 2333

With KBUILD constraints

Code defects	1835
Referential defects	439
Logical defects	299
Sum:	Σ 2573

**Result: +10%**

Summary

- Real-world system software offers **thousands of features**
 - eCos: 1,250 features
 - Linux: 12,000 features } **mostly induced by hardware!**
 - central declaration (ecosConfig, KCONFIG)
 - distributed, multi-paradigm implementation (MAKE, CPP, GCC, ...)
- This imposes great challenges for management and maintenance
 - how to ensure configurability consistency?
 - how to ensure configuration coverage?
 - how to keep pace with the constant feature increase?
- A strong call for adequate tool support ↪ **VAMOS**
 - already found **thousands** and fixed **hundreds** of defects and bugs
 - more to come!



- [1] Thorsten Berger, Steven She, Krzysztof Czarnecki, et al. *Feature-to-Code Mapping in Two Large Product Lines*. Tech. rep. University of Leipzig (Germany), University of Waterloo (Canada), IT University of Copenhagen (Denmark), 2010.
- [2] Christian Dietrich, Reinhard Tartler, Wolfgang Schröder-Preikschat, et al. "A Robust Approach for Variability Extraction from the Linux Build System". In: *Proceedings of the 16th Software Product Line Conference (SPLC '12)*. (Salvador, Brazil, Sept. 2–7, 2012). (To appear). New York, NY, USA: ACM Press, 2012. URL: http://www4.informatik.uni-erlangen.de/Publications/2012/dietrich_12_splc_draft.pdf.
- [3] Christian Dietrich, Reinhard Tartler, Wolfgang Schröder-Preikschat, et al. "Understanding Linux Feature Distribution". In: *Proceedings of the 2nd AOSD Workshop on Modularity in Systems Software (AOSD-MISS '12)*. (Potsdam, Germany, Mar. 27, 2012). Ed. by Christoph Borchert, Michael Haupt, and Daniel Lohmann. New York, NY, USA: ACM Press, 2012. ISBN: 978-1-4503-1217-2. DOI: [10.1145/2162024.2162030](https://doi.org/10.1145/2162024.2162030).
- [4] Sarah Nadi and Richard C. Holt. "Mining Kbuild to Detect Variability Anomalies in Linux". In: *Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR '12)*. (Szeged, Hungary). Ed. by Tom Mens, Yiannis Kanellopoulos, and Andreas Winter. To appear. Washington, DC, USA: IEEE Computer Society Press, 2012.



- [5] Julio Sincero, Reinhard Tartler, Daniel Lohmann, et al. "Efficient Extraction and Analysis of Preprocessor-Based Variability". In: *Proceedings of the 9th International Conference on Generative Programming and Component Engineering (GPCE '10)*. (Eindhoven, The Netherlands). Ed. by Eelco Visser and Jaakko Järvi. New York, NY, USA: ACM Press, 2010, pp. 33–42. ISBN: 978-1-4503-0154-1. DOI: 10.1145/1868294.1868300.
- [6] Reinhard Tartler, Daniel Lohmann, Christian Dietrich, et al. "Configuration Coverage in the Analysis of Large-Scale System Software". In: *ACM SIGOPS Operating Systems Review* 45.3 (Jan. 2012), pp. 10–14. ISSN: 0163-5980. DOI: 10.1145/2094091.2094095.
- [7] Reinhard Tartler, Daniel Lohmann, Julio Sincero, et al. "Feature Consistency in Compile-Time-Configurable System Software: Facing the Linux 10,000 Feature Problem". In: *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2011 (EuroSys '11)*. (Salzburg, Austria). Ed. by Christoph M. Kirsch and Gernot Heiser. New York, NY, USA: ACM Press, Apr. 2011, pp. 47–60. ISBN: 978-1-4503-0634-8. DOI: 10.1145/1966445.1966451.

