Feature Consistency in Compile-Time–Configurable System Software
Facing the Linux 10000 Feature Problem

Reinhard Tartler, Daniel Lohmann, Julio Sincero, Wolfgang Schröder-Preikschat

System Software Group
Friedrich-Alexander University
Erlangen-Nuremberg

September 7, 2011
Linux has become incredibly configurable
Configuration Complexity

- Linux has become incredibly configurable
- Complexity increases considerably
Linux has become incredibly configurable

Complexity increases considerably

Source of **bugs**!
Linux v3.0 contains:

7,702 Features
893 Kconfig files
31,281 Source files
88,897 #ifdef blocks
The Problem

Configuration

Implementation
The Problem

Source of Inconsistencies!
Finding Bugs with Tools for Static Analysis

- Bugs in **declaration and implementation**

- Excellent tool support for **static analysis**:
  - Coccinelle: Faults in Linux: Ten Years Later (ASPLOS’11)
  - Dingo: Taming Device Drivers (EuroSys’09)
  - KLEE: Automatic generation of high-coverage tests (EuroSys’08)
  - RWset: Attacking path explosion (TACAS’08)
  - EXE: Automatically generating inputs of death (CCS’06)
  - ...

R. Tartler
Feature Consistency in Compile-Time–Configurable System Software (September 7, 2011)
Finding Bugs with Tools for Static Analysis

Bugs in **declaration and implementation**

Excellent tool support for **static analysis**:

- Coccinelle: Faults in Linux: Ten Years Later (ASPLOS’11)
- Dingo: Taming Device Drivers (EuroSys’09)
- KLEE: Automatic generation of high-coverage tests (EuroSys’08)
- RWset: Attacking path explosion (TACAS’08)
- EXE: Automatically generating inputs of death (CCS’06)
- ...

Each of them check a **single** configuration:
Symbolic Inconsistency

```c
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

R. Tartler Feature Consistency in Compile-Time–Configurable System Software (September 7, 2011)
Symbolic Inconsistency

```c
static int hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu) {
    switch (action) {
    case CPU_UP_PREPARE:
    case CPU_UP_PREPARE_FROZEN:
        // [...]
    case CPU_UP_CANCELED:
    case CPU_UP_CANCELED_FROZEN:
        free_cpumask_var(cfd->cpumask);
        break;
    #ifdef CONFIG_CPU_HOTPLUG
        case CPU_DEAD:
        case CPU_DEAD_FROZEN:
            #endif
    }
    return NOTIFY_OK;
```

### Example Code

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

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

```c
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:
            // [...]
        #endif
    
        case CPU_DEAD:
        case CPU_DEAD_FROZEN:
            free_cpumask_var(cfd->cpumask);
            break;
    
    #endif
    }
    return NOTIFY_OK;
}
```

Result: Fix for a critical bug
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfn_to_mid(...)
#endif

#ifndef CONFIG_MEM
   // Block2
   #ifdef CONFIG_NUMA
      // Block3
   # else
      // Block3
   #endif
#endif
Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore
Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore

Result: code cleanup
General Approach

```
#ifdef CONFIG_DISCONTIGMEM
    // Block 1
    static ... int pfn_to_mid(...)
#endif
#ifdef CONFIG_NUMA
    // Block 2
#else
    // Block 3
#endif
```

---

R. Tartler Feature Consistency in Compile-Time–Configurable System Software (September 7, 2011)
General Approach

```c
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfn_to_mid(...)
#endif
#endif

#ifdef CONFIG_NUMA
   // Block2
   # else
      // Block3
   # endif
# endif
```
General Approach

\[ C = (FLATMEM \rightarrow MEMORY\_MODEL) \]
\[ \land (DISCONTIGMEM \rightarrow MEMORY\_MODEL) \]
\[ \land (SPARSEMEM \rightarrow MEMORY\_MODEL) \]
\[ \land (NUMA \rightarrow MEMORY\_MODEL) \]
\[ \land (DISCONTIGMEM \rightarrow NUMA) \]

```c
#ifdef CONFIG_DISCONTIGMEM
   // Block1
   static ... int pfnto_mid(...)
#endif
#ifdef CONFIG_NUMA
   // Block2
#else
   // Block3
#endif
#endif
```
General Approach

\[
C = (FLATMEM \rightarrow \text{MEMORY\_MODEL}) \\
\wedge (DISCONTIGMEM \rightarrow \text{MEMORY\_MODEL}) \\
\wedge (\text{SPARSEMEM} \rightarrow \text{MEMORY\_MODEL}) \\
\wedge (\text{NUMA} \rightarrow \text{MEMORY\_MODEL}) \\
\wedge (\text{DISCONTIGMEM} \rightarrow \text{NUMA})
\]

\[
I = (\text{Block}_1 \leftrightarrow \text{DISCONTIGMEM}) \\
\wedge (\text{Block}_2 \leftrightarrow \text{Block}_1 \wedge (\text{NUMA}) \\
\wedge (\text{Block}_3 \leftrightarrow \text{Block}_1 \wedge \neg \text{Block}_2)
\]
General Approach

Crosscheck both formulas with a SAT solver:

\[
\text{dead}? \, = \, sat(C \wedge I \wedge Block_N) \\
\text{undead}? \, = \, sat(C \wedge I \wedge \neg Block_N \wedge parent(Block_N))
\]
Implementation Challenges

Accuracy

- Conceptually *no false positives*
- **Exact** identification of variation points
Implementation Challenges

- **Accuracy**
  - Conceptually no false positives
  - Exact identification of variation points

- **Coverage**
  - Extract configuration model for all 22 architectures
  - Defect detected on each architecture
Implementation Challenges

- **Accuracy**
  - Conceptually no false positives
  - Exact identification of variation points

- **Coverage**
  - Extract configuration model for all 22 architectures
  - Defect detected on each architecture

- **Performance**
  - Easy and fast to use during incremental builds
  - Possible by problem slicing
  - Complete run on Linux in less than 10 minutes
## Results

<table>
<thead>
<tr>
<th>subsystem</th>
<th>#ifdefs</th>
<th>logic</th>
<th>symbolic</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/</td>
<td>33757</td>
<td>345</td>
<td>581</td>
<td>926</td>
</tr>
<tr>
<td>drivers/</td>
<td>32695</td>
<td>88</td>
<td>648</td>
<td>736</td>
</tr>
<tr>
<td>fs/</td>
<td>3000</td>
<td>4</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>include/</td>
<td>7241</td>
<td>6</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>kernel/</td>
<td>1412</td>
<td>7</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>mm/</td>
<td>555</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>net/</td>
<td>2731</td>
<td>1</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>sound/</td>
<td>3246</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>virt/</td>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>other subsystems</td>
<td>601</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>∑</strong></td>
<td>85291</td>
<td>460</td>
<td>1316</td>
<td>1776</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>fix proposed</th>
<th>confirmed defect</th>
<th>confirmed rule-violation</th>
<th>pending</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>150 (1)</td>
<td>214 (22)</td>
<td>364 (23)</td>
<td>101 (0)</td>
</tr>
<tr>
<td></td>
<td>38 (1)</td>
<td>116 (20)</td>
<td>154 (21)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>88 (0)</td>
<td>21 (2)</td>
<td>109 (2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24 (0)</td>
<td>77 (0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
#ifdef

<table>
<thead>
<tr>
<th>subsystem</th>
<th>ifdef</th>
<th>logic</th>
<th>symbolic</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/</td>
<td>33757</td>
<td>345</td>
<td>581</td>
<td>926</td>
</tr>
<tr>
<td>drivers/</td>
<td>32695</td>
<td>88</td>
<td>648</td>
<td>736</td>
</tr>
<tr>
<td>fs/</td>
<td>3000</td>
<td>4</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>include/</td>
<td>7241</td>
<td>6</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>kernel/</td>
<td>1412</td>
<td>7</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>mm/</td>
<td>555</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>net/</td>
<td>2731</td>
<td>1</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>sound/</td>
<td>3246</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>virt/</td>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>other subsystems</td>
<td>601</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>∑</td>
<td>85291</td>
<td>460</td>
<td>1316</td>
<td>1776</td>
</tr>
</tbody>
</table>

| fix proposed | 150 (1) | 214 (22) | 364 (23) |
| confirmed defect | 38 (1) | 116 (20) | 154 (21) |
| confirmed rule-violation | 88 (0) | 21 (2) | 109 (2) |
| pending | 24 (0) | 77 (0) | 101 (0) |

We have found **1776** configurability issues

Submitted **123** patches for **364** defects

**20** are confirmed **new bugs** (affecting binary code)

Cleaned up **5129** lines of cruft code
Further Application: Configuration Coverage

- Current ongoing work, accepted at PLOS’11

- Configuration Coverage is defined as:
  fraction of selected configuration-conditional blocks
divided by the number of available configuration-conditional blocks.

- How to catch bugs that apply only on specific kernel configurations?
  ⇒ Test them on as many configurations as possible

- Static analyzers (sparse, smatch, ...) scan a particular kernel configuration
  ⇒ How to efficiently expand their coverage?
Historical analysis of allyes coverage

- allyesconfig blocks (n)
- total blocks (n)
- files (n)
- coverage allyesconfig (%)

Graph showing the evolution of allyes coverage across different versions from v2.6.22 to v3.0-rc2.
Concrete Example

```c
#ifdef CONFIG_DISCONTIGMEM
static inline int pfn_to_nid(unsigned long pfn)
{
#ifdef CONFIG_NUMA
    return ((int) physnode_map[(pfn) / PAGES_PER_ELEMENT]);
#else
    return 0;
#endif
#else
    return 0;
#endif
}
#endif
```

Possible Configurations:

- Neither, DISCONTIGMEM, DISCONTIGMEM ∧ NUMA

Additionally testing the configuration NUMA does not increase the Configuration Coverage.
Realization

KConfig Files

config HOTPLUG CPU
  bool "Support for ..."
  depends on SMP & & ...
Implementation Challenges

- Proper extraction of Configurations constraints
  - Kconfig (implemented in undertaker)
  - Kbuild constraints (largely unhandled)

- Expansion of Partial Configurations
  - Naïve approach has some surprising effects (i.e., fails sometimes)
  - Kconfig-sat seems promising, but unfortunately discontinued
# Evaluation

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzed files</td>
<td>10,365</td>
</tr>
<tr>
<td>Files with variability</td>
<td>3,163</td>
</tr>
<tr>
<td>Rate of files with variability</td>
<td>30.52%</td>
</tr>
<tr>
<td>Sum of all (partial) configurations</td>
<td>4,435</td>
</tr>
<tr>
<td>Sum of configuration controlled conditional blocks</td>
<td>16,444</td>
</tr>
<tr>
<td>Sum of blocks selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Sum of all blocks selected by undertaker-coverage</td>
<td>13,844</td>
</tr>
<tr>
<td>Coverage allyesconfig (non-dead-corrected)</td>
<td>70.00%</td>
</tr>
<tr>
<td>Coverage undertaker (non-dead-corrected)</td>
<td>84.19%</td>
</tr>
<tr>
<td>Dead blocks</td>
<td>1,778</td>
</tr>
<tr>
<td>Selectable blocks (excluding dead blocks)</td>
<td>14,666</td>
</tr>
<tr>
<td>Selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Covered by undertaker</td>
<td>13,844</td>
</tr>
<tr>
<td>allyesconfig coverage</td>
<td>78.49%</td>
</tr>
<tr>
<td>undertaker coverage</td>
<td>94.40%</td>
</tr>
<tr>
<td>undertaker coverage / allyesconfig coverage</td>
<td>1.20</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzed files</td>
<td>10,365</td>
</tr>
<tr>
<td><strong>Files with variability</strong></td>
<td>3,163</td>
</tr>
<tr>
<td>Rate of files with variability</td>
<td>30.52%</td>
</tr>
<tr>
<td><strong>Sum of all (partial) configurations</strong></td>
<td>4,435</td>
</tr>
<tr>
<td>Sum of <em>configuration controlled</em> conditional blocks</td>
<td>16,444</td>
</tr>
<tr>
<td>Sum of blocks selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Sum of all blocks selected by undertaker-coverage</td>
<td>13,844</td>
</tr>
<tr>
<td>Coverage allyesconfig (non-dead-corrected)</td>
<td>70.00%</td>
</tr>
<tr>
<td>Coverage undertaker (non-dead-corrected)</td>
<td>84.19%</td>
</tr>
<tr>
<td>Dead blocks</td>
<td>1,778</td>
</tr>
<tr>
<td>Selectable blocks (excluding dead blocks)</td>
<td>14,666</td>
</tr>
<tr>
<td>Selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Covered by undertaker</td>
<td>13,844</td>
</tr>
<tr>
<td>allyesconfig coverage</td>
<td>78.49%</td>
</tr>
<tr>
<td>undertaker coverage</td>
<td>94.40%</td>
</tr>
<tr>
<td>undertaker coverage / allyesconfig coverage</td>
<td>1.20</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzed files</td>
<td>10,365</td>
</tr>
<tr>
<td><strong>Files with variability</strong></td>
<td>3,163</td>
</tr>
<tr>
<td>Rate of files with variability</td>
<td>30.52%</td>
</tr>
<tr>
<td><strong>Sum of all (partial) configurations</strong></td>
<td>4,435</td>
</tr>
<tr>
<td>Sum of configuration controlled conditional blocks</td>
<td>16,444</td>
</tr>
<tr>
<td>Sum of blocks selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Sum of all blocks selected by undertaker-coverage</td>
<td>13,844</td>
</tr>
<tr>
<td><strong>Coverage allyesconfig (non-dead-corrected)</strong></td>
<td>70.00%</td>
</tr>
<tr>
<td><strong>Coverage undertaker (non-dead-corrected)</strong></td>
<td>84.19%</td>
</tr>
<tr>
<td>Dead blocks</td>
<td>1,778</td>
</tr>
<tr>
<td>Selectable blocks (excluding dead blocks)</td>
<td>14,666</td>
</tr>
<tr>
<td>Selected by allyesconfig</td>
<td>11,511</td>
</tr>
<tr>
<td>Covered by undertaker</td>
<td>13,844</td>
</tr>
<tr>
<td>allyesconfig coverage</td>
<td>78.49%</td>
</tr>
<tr>
<td>undertaker coverage</td>
<td>94.40%</td>
</tr>
<tr>
<td>undertaker coverage / allyesconfig coverage</td>
<td>1.20</td>
</tr>
</tbody>
</table>
### Evaluation

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzed files</td>
<td>10,365</td>
</tr>
<tr>
<td><strong>Files with variability</strong></td>
<td>3,163</td>
</tr>
<tr>
<td>Rate of files with variability</td>
<td>30.52%</td>
</tr>
<tr>
<td><strong>Sum of all (partial) configurations</strong></td>
<td>4,435</td>
</tr>
<tr>
<td>Sum of configuration controlled conditional blocks</td>
<td>16,444</td>
</tr>
<tr>
<td>Sum of blocks selected by <code>allyesconfig</code></td>
<td>11,511</td>
</tr>
<tr>
<td>Sum of all blocks selected by <code>undertaker-coverage</code></td>
<td>13,844</td>
</tr>
<tr>
<td><strong>Coverage <code>allyesconfig</code> (non-dead-corrected)</strong></td>
<td>70.00%</td>
</tr>
<tr>
<td><strong>Coverage <code>undertaker</code> (non-dead-corrected)</strong></td>
<td>84.19%</td>
</tr>
<tr>
<td>Dead blocks</td>
<td>1,778</td>
</tr>
<tr>
<td>Selectable blocks (excluding dead blocks)</td>
<td>14,666</td>
</tr>
<tr>
<td>Selected by <code>allyesconfig</code></td>
<td>11,511</td>
</tr>
<tr>
<td>Covered by <code>undertaker</code></td>
<td>13,844</td>
</tr>
<tr>
<td><code>allyesconfig</code> coverage</td>
<td>78.49%</td>
</tr>
<tr>
<td><code>undertaker</code> coverage</td>
<td>94.40%</td>
</tr>
<tr>
<td><code>undertaker</code> coverage / <code>allyesconfig</code> coverage</td>
<td>1.20</td>
</tr>
</tbody>
</table>
Evaluation

**Analyzed files**: 10,365

**Files with variability**: 3,163

Rate of files with variability: 30.52%

**Sum of all (partial) configurations**: 4,435

With **30 percent** more compiler calls (static analysis runs)

We get **15 percent** more Configuration Coverage

<table>
<thead>
<tr>
<th>Covered by undertaker</th>
<th>13,844</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>allyesconfig coverage</strong></td>
<td>78.49%</td>
</tr>
<tr>
<td><strong>undertaker coverage</strong></td>
<td>94.40%</td>
</tr>
<tr>
<td>undertaker coverage / allyesconfig coverage</td>
<td>1.20</td>
</tr>
</tbody>
</table>
Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect

- **Configuration** and **implementation** need to be kept consistent

- Configuration Coverage increases the effectiveness of existing tools.
Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect.

- Configuration and implementation need to be kept consistent.

- Configuration Coverage increases the effectiveness of existing tools.

Vision:

- Explorative tool for visualizing and checking Variability in Kconfig and realization.

- Linux Feature Explorer (LIFE)
Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect

- **Configuration and implementation** need to be kept consistent

- Configuration Coverage increases the effectiveness of existing tools.

**Vision:**

- **Exploratory tool for visualizing and checking** Variability in Kconfig and realization

- Linux Feature Explorer (**LIFE**)

http://vamos.informatik.uni-erlangen.de/trac/undertaker