A User Survey of Configuration Challenges in Linux and eCos

Arnaud Hubaux
PReCISE Research Centre,
University of Namur,
Belgium
ahu@info.fundp.ac.be

Yingfei Xiong
Generative Software
Development Lab,
University of Waterloo,
Canada
y6xiong@gsd.uwaterloo.ca

Krzysztof Czarnecki
Generative Software
Development Lab,
University of Waterloo,
Canada
kczarnec@gsd.uwaterloo.ca

ABSTRACT

Operating systems expose sophisticated configurability to handle variability in hardware platforms like mobile devices, desktops, and servers. The variability model of an operating system kernel like Linux contains thousands of options guarded by hundreds of complex constraints. To guide users throughout the configuration and ensure the validity of their decisions, specialized tools known as configurators have been developed. Despite these tools, configuration still remains a difficult and challenging process. To better understand the challenges faced by users during configuration, we conducted two surveys, one among Linux users and another among eCos users. This paper presents the results of the surveys along three dimensions: configuration practice; user guidance; and language expressiveness. We hope that these results will help researchers and tool builders focus their efforts to improve tool support for software configuration.

1. INTRODUCTION

An important aspect of variability modeling is configuration. In a configuration process, the user assigns values to options to produce a configuration. To help users produce a configuration efficiently and correctly, researchers and practitioners have developed tools to support the configuration process. These tools are called *configurators*. Typically, configurators assist users in two aspects. First, configurators present the options and their explanations as defined in a variability model—such as a feature model [14] or a decision model [23]—, and help users navigate in the model. Second, configurators ensure the validity of values assigned to the options, i.e., that the values conform to the types of the options, the hierarchical constraints of the variability model, and the cross-tree constraints on the variability model.

Various configurators have been developed for different variability languages. For example, Linux is equipped with different configurators for configuring the kernel, according to variability models expressed in the Kconfig language. Similarly, eCos, an embedded operating system, comes with two configurators for its CDL language. In the research community, different open source [1, 3, 15, 17] and commercial [5, 6] configurators have been developed for feature models. Similar tools for decision models also exist [9].

Generally, configurators can be classified into command-line-based and GUI-based. Figure 1 shows a screenshot of a GUI-based configurator: eCos *configtool* for the CDL language. On the left panel, the options are organized into a tree according to their definitions in the CDL language. Also, different input methods (such as check- and textboxes) ensure that the input values conform to the types of the options. When constraints are violated, configtool either automatically disables some options to prevent configuration errors (such as option *Use big-endian mode*) or reports errors. Command-line-based tools provide similar functionalities, in a less interactive but scriptable way.

Although sophisticated configurator support exists, configuring systems according to a variability model still remains a difficult process. Users often cannot complete a configuration without external help [2] and configuration errors can cause disastrous results [27]. Thus, we need a better understanding—backed by empirical evidence—of what configuration challenges the users of modern configurators still face.

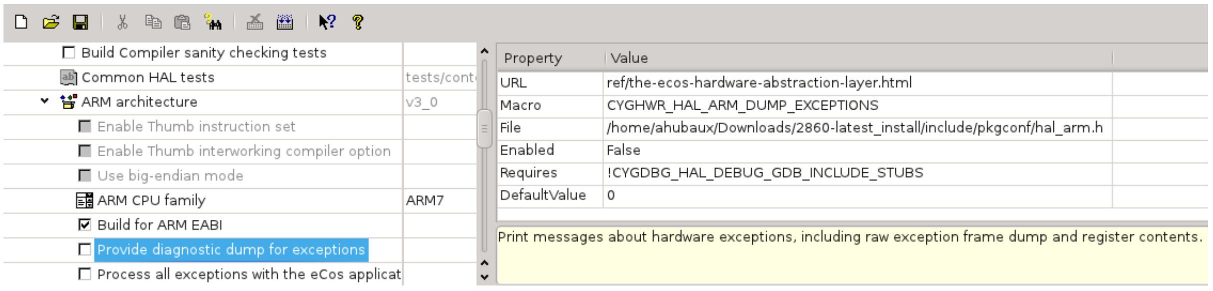
To address this need, we have conducted two online surveys with the users of the Kconfig-based configurators and the CDL-based configurators, respectively. We submitted two questionnaires to forums, mailing lists, and experts with whom we collaborate, and collected answers from 97 Linux users with up to 20 years of experience, and 9 eCos users with up to 7 years of experience.

This paper presents the results of this survey. They are grouped in three major categories:

Configuration practice. Our survey showed that the configuration of an operating system is mostly a single user activity, and that users resolve conflicting decisions as soon as they appear.

User guidance. Many respondents complained about the lack of guidance for making configuration decisions, and the low quality of the advice provided by the configurators. Specifically, activating an inactive option can be difficult, advice is often incomplete, and determining only the necessary options is hard.

Language expressiveness. The functionalities provided by the configurators largely depend on the expressiveness



```
☐ Build Compiler sanity checking tests
Common HAL tests                               tests/cont
ARM architecture                               v3_0
  Enable Thumb instruction set
  Enable Thumb interworking compiler option
  Use big-endian mode
  ARM CPU family                               ARM7
  ☑ Build for ARM EABI
  ☐ Provide diagnostic dump for exceptions
  ☐ Process all exceptions with the eCos applicat
```

Property	Value
URL	ref/the-ecos-hardware-abstraction-layer.html
Macro	CYGHWR_HAL_ARM_DUMP_EXCEPTIONS
File	/home/ahubaux/Downloads/2860-latest_install/include/pkgconf/hal_arm.h
Enabled	False
Requires	!CYGDBG_HAL_DEBUG_GDB_INCLUDE_STUBS
DefaultValue	0

Print messages about hardware exceptions, including raw exception frame dump and register contents.

Figure 1: eCos configtool

of the configuration languages. One respondent claimed that configurators should provide support for multiple but differently configured instances of the same piece of hardware, suggesting the need for multiple instantiation of features [8].

The rest of the paper is organized as follows. Section 2 introduces the languages Kconfig and CDL and their configurators. Section 3 explains how we conducted the survey and Section 4 presents the results. Section 5 discusses threats to validity. Finally, Section 6 describes related work and Section 7 concludes the paper.

2. BACKGROUND

This section provides a brief description of the two variability languages in our study, Linux Kconfig and eCos CDL, and their configurators. A more detailed description is available elsewhere [4].

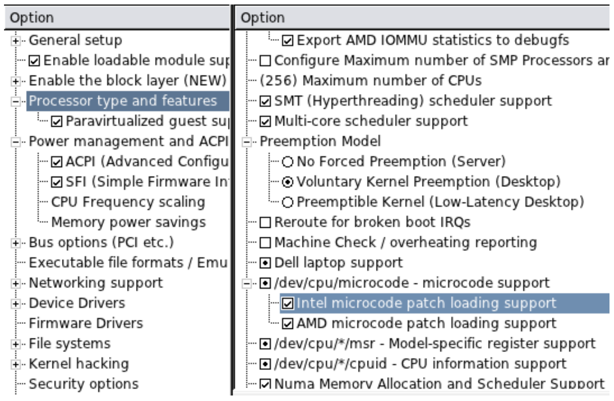
Kconfig is the language designed for describing variants in the Linux kernel. Figure 2 shows a fragment of a Kconfig model. This fragment defines 6 options, where NR_CPUS is an integer option; SCHED_MC, PREEMPT_NONE, PREEMPT_VOLUNTARY and PREEMPT are Boolean options; and I8K is a tristate option. A tristate option can take three values: true, false, or module, indicating that the corresponding functionality should be statically included into the kernel, should not be available, or should be available as a dynamically loadable module, respectively. The fragment also contains a *choice* construct that allows the selection of only one option among the three. By default, PREEMPT_NONE is selected.

To support the configuration of Kconfig models, several configurators have been developed. A popular one is *xconfig*. Figure 3 shows a screenshot of xconfig. The left panel contains the high-level menu decomposition; the right panel details the available options for each menu. For instance, the *Processor type and features* menu contains the options of the fragment in Figure 2. In the displayed configuration, the *Maximum number of CPUs* (NR_CPUS) is set to 256, the *Multi-core scheduler support* (SCHED_MC) and *Voluntary Kernel Preemption (Desktop)* (PREEMPT_VOLUNTARY) options are enabled, and the *Dell laptop support* option (I8K) is set to module.

Configuration options are often related by constraints, and violations of these constraints are considered as configuration errors. To model these constraints and prevent configuration errors, Kconfig introduces several types of constructs. One such construct is the *default* property, as in lines 6-9. If the so-called *prompt condition* following the *if* in line 3 is violated, this option is hidden from the user and the default

```
1  ...
2  config NR_CPUS
3     int "Maximum number of CPUs" if SMP && !MAXSMP
4     range 2 8 if SMP && X86_32 && !X86_BIGSMP
5     range 2 512 if SMP && !MAXSMP
6     default "1" if !SMP
7     default "4096" if MAXSMP
8     default "32" if SMP && (X86_NUMAQ || ...)
9     default "8" if SMP
10 ...
11 config SCHED_MC
12    def_bool y
13    prompt "Multi-core scheduler support"
14    depends on X86_HT
15 ...
16 choice
17    prompt "Preemption Model"
18    default PREEMPT_NONE
19 config PREEMPT_NONE
20    bool "No Forced Preemption (Server)"
21 config PREEMPT_VOLUNTARY
22    bool "Voluntary Kernel Preemption (Desktop)"
23 config PREEMPT
24    bool "Preemptible Kernel (Low-Latency Desktop)"
25 endchoice
26 ...
27 config I8K
28    tristate "Dell laptop support"
29 ...
```

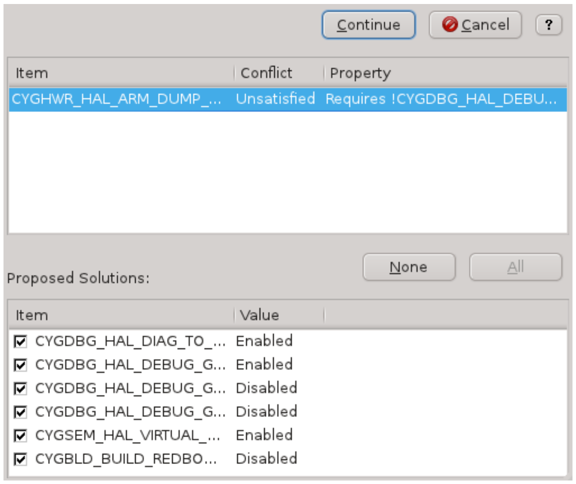
Figure 2: A fragment of a Kconfig model



```
Option
+ General setup
  ☑ Enable loadable module sup
+ Enable the block layer (NEW)
- Processor type and features
    ☑ Paravirtualized guest su
- Power management and ACPI
    ☑ ACPI (Advanced Configu
    ☑ SFI (Simple Firmware In
    CPU Frequency scaling
    Memory power savings
+ Bus options (PCI etc.)
  Executable file formats / Emu
+ Networking support
+ Device Drivers
  Firmware Drivers
+ File systems
+ Kernel hacking
  Security options
```

```
Option
    ☑ Export AMD IOMMU statistics to debugfs
  ☐ Configure Maximum number of SMP Processors ar
  (256) 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)
  ☐ Reroute for broken boot IRQs
  ☐ Machine Check / overheating reporting
  ⊡ Dell laptop support
- ⊡ /dev/cpu/microcode - microcode support
    ☑ Intel microcode patch loading support
    ☑ AMD microcode patch loading support
  ⊡ /dev/cpu/*/msr - Model-specific register support
  ⊡ /dev/cpu/*/cpuid - CPU information support
  ☑ Numa Memory Allocation and Scheduler Support
```

Figure 3: Configuring the model in xconfig



Item	Conflict	Property
CYGHWR_HAL_ARM_DUMP_...	Unsatisfied	Requires !CYGDBG_HAL_DEBU...

Proposed Solutions:

Item	Value
☑ CYGDBG_HAL_DIAG_TO_...	Enabled
☑ CYGDBG_HAL_DEBUG_G...	Enabled
☑ CYGDBG_HAL_DEBUG_G...	Disabled
☑ CYGDBG_HAL_DEBUG_G...	Disabled
☑ CYGSEM_HAL_VIRTUAL_...	Enabled
☑ CYGBLD_BUILD_REDBO...	Disabled

Figure 4: Fix generation in configtool

value matching the first satisfied constraint in lines 6-9 is automatically assigned. Kconfig also introduces other advanced properties such as the *depends on* construct, which imposes dependencies, and *range*, which restricts the possible values a variable can take. Essentially, Kconfig-based configurators use *decision propagation* to deactivate unavailable options and prevent configuration errors.

CDL is the variability modeling language for the eCos operating system. It works in a similar way as Kconfig. First, the developers describe the configuration options and constraints in CDL as a variability model; they also implement the effect of the options in the code using preprocessor directives. Then, users configure the variability model and obtain a customized operating system. As mentioned in Section 1, two configurators are available for CDL models: a GUI-based configurator *configtool* and a commandline configurator *ecosconfig*.

CDL also employs a construct similar to the prompt condition in Kconfig: *active-if*. When an active-if constraint is violated, the corresponding option is disabled and its value is automatically considered as zero (like option *Use big-endian mode* in Figure 1). This behavior is an application of decision propagation. Besides active-if constraints, CDL allows another type of constraints: *requires* constraint. When a requires constraint is violated, the configurator reports an error. For instance, when the user enables *Provide diagnostic dump for exceptions* (`CYGHWR_HAL_ARM_DUMP_EXCEPTIONS`), the configurator reports an error because the requires constraint is not satisfied.

To help users resolve configuration errors, configtool generates fixes for configuration errors. A fix is a concrete change on the configuration to satisfy the corresponding requires constraint. For the configuration error, configtool will propose the fix shown in Figure 4. If the user accepts this fix, all the options under "*Proposed solutions:*" are changed accordingly, and the configuration error will be fixed. eCos thus adds *conflict resolution* to decision propagation. Note that the list of options to change represents a single fix to the configuration. Even though multiple fixes may exist for a single configuration error, the current version of configtool proposes at most one fix.

3. METHODOLOGY

The research question addressed in this paper is: *What are the challenges faced by Linux and eCos users during configuration?*

To answer this question, we conducted two online surveys among Linux and eCos users, respectively. The specific goals of the surveys were to better understand:

- the *profile* of the respondent (e.g., what is his level of expertise, and what type of projects he is using the operating system for);
- the *configuration practice* of the respondent (e.g., how many users are involved in the configuration, and how many changes to the default configuration are made per project);
- the *challenges* faced during configuration (e.g., which configuration task is the most difficult, and how the configuration problems are solved).

The questions in the survey contained dichotomous and multichotomous choices, rating scales to collect interval levels, and open text fields to collect explanations and examples. The complete Linux and eCos questionnaires are available online.[1]

To obtain a representative panel of Linux users, the Linux survey was posted on 11 forums and mailing lists of mainstream Linux distributions, and on the Linux Kernel mailing list. The complete list of distribution forums used can be found elsewhere [11]. The eCos community is much smaller; the survey was posted on the developer and discussion eCos mailing lists, and submitted to the developers of several projects, including ReconOS and DancOS.

In total, 97 Linux users filled out the survey. The results presented in this paper are based on their answers and on discussions on a Gentoo forum and the opensuse-factory mailing list, following our posts. The quantitative data reported here is based on a subset of 81 answers. 16 answers were discarded either because the responses to some questions were inappropriate or too vague to conduct quantitative or qualitative analyses.

Roughly half (49%) of these Linux users claimed to be experts, with up to 20 years of experience with Linux. Most respondents have experience with more than one distribution (three on average). Overall, 39 distributions have been reported, showing a great diversity in the background of the respondents. The use of Linux was primarily for personal purpose, followed by server maintenance, system administration, development, embedded systems, and virtual machines. The users who indicated personal use specified frequently other uses too (e.g., server or enterprise). This pattern is reasonable as many people use Linux both personally and professionally.

A total of 9 eCos users answered the survey, and no answers were excluded from further analysis. A third (33%) of the respondents consider themselves expert, counting up to 7 years of experience with eCos. The types of projects include FPGA (Field-Programmable Gate Array), solar inverter, powerline management, and research.

The results of the eCos and Linux surveys are presented in the next section. The quantitative results were obtained through categorizations, interval comparisons, additions, and

[1]See `http://gsd.uwaterloo.ca/GSDLAB-TR2011-09-29`

simple statistical functions (max, min, average and median). The qualitative analysis is based on comments from 43 participants, and 2 forum and mailing list members.

4. RESULTS

We now report our results, classifying them into two types. First, findings backed up by solid evidence are formulated as statements (**boldfaced**). Secondly, emerging results that still require further investigation are formulated as research questions (*italicized*). All results are organized in three categories: configuration practice, user guidance, and language expressiveness. The quotations used to illustrate and motivate the findings and research questions are verbatim copies of comments made by the respondents. The only edits we made is hiding irrelevant text, indicated by ellipses [. . .], and clarifications, enclosed in brackets [].

4.1 Configuration practice

Configuring an operating system is mostly a single user activity. For both Linux and eCos, almost 90% of the respondents report that they configure the kernel alone. When configuration is collaborative, very few people are involved (5 at most for Linux and 2 at most for eCos). When conflicts occur between decisions, they are resolved either by consensus or by the project leader. Half of the Linux respondents make between 20 and 50 changes to a default configuration. In extreme cases, some users reported making more than 2000 changes. As for eCos, the respondents make between 10 and 20 changes to a default configuration on average, with a maximum of 50.

Conflicting decisions are resolved as soon as they appear. The eCos configurator allows users to postpone conflict resolution, that is, users can "live with inconsistencies" until they decide to resolve them. Some authors have advocated this practice (e.g. [21]). Yet, all the respondents to the eCos study answered that they solve conflicts as soon as they occur. This observation could not be corroborated for Linux because all the Kconfig configurators prevent conflicts through decision propagation.

Are two profiles of users emerging? Modern Linux distributions now render the configuration of the Linux kernel unnecessary for most users. A limited knowledge of operating systems is sufficient to install and administer a personal computer. Those who still configure the kernel have specific needs (e.g., server or virtual machine reconfiguration and driver development) that require a significant amount of experience and knowledge. One respondent explained:

> I really think the kernel has a large number of users of which only very few use the configurable system since the modularization allowed distributions to deliver a kernel in such a way that no configuration is needed by end users anymore. Thus, the users of the system are mainly the developers of the kernel and the packager of the kernel packages in the distributions.

4.2 User guidance

Activating an inactive option is difficult. When a user wants to change an option, but this option is hidden/disabled because the corresponding prompt/active-if conditions are violated, the user needs to first satisfy the constraint to make the option visible/enabled. This process is called the *activation of an inactive option.* 78% of the eCos respondents report that activating an inactive option is the most difficult task; setting the value of an attribute is the most difficult task for the remaining respondents. 56% of the respondents consider that enabling/disabling an option is actually a problem in practice. 20% of the Linux users report that, when they need to change an inactive option, they need at least a "few dozen minutes", on average, to figure out how to activate it. Furthermore, to activate an inactive option users either rely on their own expertise (26%) or, most frequently, read and follow manually the constraints described in the documentation (46%). 13 respondents provided a more elaborate response to this question. Two of them phrased it as follows:

> As far as consistency checking and helping determine inter-related dependencies on settings, I have long wished for a better kernel configuration tool [. . .], but it seems that the kernel guys learn their way around the configurator by much exposure, and the rest of us have to just figure it out [. . .]

> I'd like all driver names mentioned in help, for instance: enabling this driver as will give you driver foo.

Advice is often incomplete, hard to understand or incorrect. 18 Linux users complained about the incorrect, incomplete, and unclear advice provided by documentation or the tool. These flaws affect their efficiency during the configuration of the kernel, e.g.:

> Many options' effects are insufficiently documented so I end up blindly choosing the default or recommended value.

> Quite often there is a recommended option [. . .] which is invaluable when you don't know what to choose, but there are also some items where you just have to take your best guess. This can be particularly difficult for a first-timer. With more experience, you sort of get a feel for what's important and what isn't.

> [. . .] the defaults often disagree with the help text advice. For example, it will say, "If not sure, say Y", but the option will be disabled by default, or vice-versa.

> My most common problem is inconsistencies in the Kconfig files.

> Stupid/nonsensical/dangerous defaults.

The conflict resolution fixes proposed by the configurator are incomplete. Existing eCos configurators generate only one fix to resolve a conflict. However, there are often multiple solutions to resolving a conflict, and the user may prefer other solutions. 7 out of 9 eCos users have encountered situations where the *generated fix is not useful.* That claim is corroborated by Berger *et al.* [4] who report that eCos users complain about the incompleteness of fixes on the mailing list.

Determining only the necessary options is hard. Overall, the size of the Linux kernel grows with every release [16]. This growing complexity often leads to bloated, one-size-fits-all kernel configurations. These kernels are usually fine to boot mainstream distribution but are often too general for dedicated machines. The selection of the desired options is further complicated by the poor support for searches for options and minimal configurations. Some respondents formulated these challenges as follows:

> It makes sense [...] to get something up and running to start the installation process, but a production kernel needs to aviod unneeded components both for speed, and economy reasons, not to mention security.
>
> Minimising the number of modules compiled in is important in order to reduce the potential "attack surface" within the kernel. Thus, enabling all options is not an option, and discovering exactly which options are required for given hardware is necessary.
>
> Usually it is difficult to search for new options. In the nconfig interface, using F8 one can search for a config option, but the search term must be close enough to (a substring of) the actual configuration/module name. Till now, to determine the options I need to enable, I usually follow a sequence of steps (1) Feed lspci -n into a debian hcl [hardware compatibility list] webpage which returns me the list of modules I need; (2) Search for those modules in the kernel; (3) Search in Google or Gentoo documentation for any other options I might want to enable [...] it would be much nicer if the kernel provided a way to search through all the documentation [...]

The recent *kernel seeds*[2] initiative is a first step toward more economical kernels. The goal of a kernel seed is to include a minimum amount of options for the kernel to be bootable. Users then only have to tweak the options specific to their equipment (e.g., graphics driver or file system).

Is better support to handle kernel updates necessary? Four Linux users complained about the difficulty to maintain a configuration across updates. Tools like `make oldconfig` help avoid reselecting options during version upgrades. However, more information is necessary to track and understand the impact of changes:

> Occasionally, options which were introduced or changed in new kernel versions aren't fully documented, giving insufficient information regarding whether I should enable or disable the new option.
>
> Driver cleanups that move PCI ids between PCIe and PCI drivers can be a real pain if it happens on a kernel update and I wasn't expecting it.

Is community-based configuration problem solving necessary? One expert Linux user suggests the creation of a knowledge base of known problems and solutions. A case-based reasoning expert system would then be used to match the posted problems to existing problems and solutions. Applied to configuration, this would mean that configurations are shared and can be used to guide users during their configuration process. Bad configuration should also be posted, provided a description of the problem is also available. The expert proposes to design the solution in JESS (Java Expert System Shell), which integrates easily with the web (Apache Tomcat). A web-based interface would be ideal as it would allow the global Linux community to contribute to it easily. Furthermore, the knowledge base could already be pre-loaded with data mined from forum threads that archive a lot of human dialog.

[2]`http://www.kernel-seeds.org/`

4.3 Language expressiveness

Should option multiplicities be included? The need for multiplicities has already been debated in variability modelling [8, 20]. One eCos user expressed the need to include this functionality in configurators:

> I had an issue whereby I added a SPI flash driver in order to support a flash device. To get one flash device supported was easy. You checked the boxes and the flash device popped up and was usable. I had [...] another of the same flash device on a different chip [...] and when I tried to add the second device the CDL tool fell apart. I had to hack the source CDL files of the [...] flash driver to make this possible.

This quotation gives a clear example of a use case for multiple instantiation. The respondent wants to include multiple instances of a flash driver, each configured differently, and eCos does not provide this functionality. CDL allows to capture the variability of *one* particular piece of equipment. It does not allow to enable or disable multiple instances of the same equipment on a single board.

5. THREATS TO VALIDITY

We see two major external threats to this work. First, the results in this paper are specific to Linux and eCos. Yet, both Linux and eCos tools use configuration strategies that are also used in other tools; further, one of them, decision propagation, is shared between the Linux and eCos tools. Moreover, we observed a significant overlap between the challenges reported by users of both operating systems. Also, the Linux kernel is a configurable system with a large community of users and developers and spans a wide variety of application domains.

Secondly, the respondents may not be representative of the user population, i.e., they are not a controlled random sample from the user population. The small sample of eCos respondents (9) is counterbalanced by the significant sample (97 respondents) for Linux, meaning that the results have merit. Also, the different levels of expertise, years of experience, types of projects, and high number of used Linux distributions (39) show a great diversity in the background and profile of the respondents.

Two main internal threats also exist. First, our survey might reflect a partial picture of the configuration practice. Specifically, our formulation of the questions and the explicit description of possible problems like option activation and attribute assignments might have influenced the respondents. To diminish that threat, we collaborated with both eCos and Linux users prior to writing the questions in the survey to gain some understanding of their practice. We also capitalized on our own experience with the Linux kernel to pinpoint likely challenges.

Secondly, we might have misinterpreted some answers and comments made by the respondents. To mitigate that threat, we followed up on some questions with the respondents via email and through discussions on forums and mailing lists. We used the received answers to clarify ambiguities and go deeper into some points raised in the original responses.

6. RELATED WORK

Both CDL and Kconfig fall under the umbrella of variability modelling languages [7]. Another popular variability modelling language is feature models [14]. Feature models have expressiveness comparable to that of Kconfig [4, 24] and CDL [4], and have been successfully implemented in commercial product-line tools like pure::variants [5] and Gears [6]. Decision propagation for feature models has been studied extensively [12,13,18,19] and conflict resolution techniques are emerging [25,26].

Rabiser et al. [22] conducted a systematic literature survey and an expert survey on product-derivation support (aka product-configuration support), that is, support for the selection and customization of assets from a product-line to create individual products. About 20% of the analyzed papers related to product derivation dealt with this topic as their primary one. The challenges discussed in these papers include low degree of automation in product derivation, missing support for project management, and visualization being inadequate for users. In the expert survey, conducted among product-line engineering researchers and practitioners attending VaMoS'08 and SPLC'08, respondents rated tool flexibility and adaptability, support for managing product-specific requirements requiring custom development, and flexible visualizations as significantly more important than guidance for decision making and project management support. Our survey gives a different perspective on product configuration as it is focused on the users of actual derivation tools. In contrast to the surveyed experts, the users do care about the guidance for decision making and complain about it if it is inadequate; however, both the users and the experts agree on the importance to support users in the exploration of the available configuration options via searches, filtering, and visualization. Further, additional requirements suggested by the experts included "understandable constraint resolution or guidance in case of problems," which also coincides with the needs of the users in our survey.

In a preliminary review on the application of feature models in practice, Hubaux *et al.* [10] conclude that evidence on the practical use of such models is not well documented. In an initial pool of 414 papers related to product-line engineering, only 16 papers focussed on variability models and industrial practice. Only 8 of these 16 papers discussed actual cases of using feature models in industrial projects and none showed examples of such models. This finding indicates that challenges related to the use of variability models might be hard to extract from the research community (note that feature models are the most popular form of variability modeling today, both in the number of published papers and available tools). Thus, we aim at collecting challenges related to configurators used in mainstream open-source projects and contributing these back to the variability modelling community.

Yin *et al.* [27] studied 546 real world configuration errors from commercial and open source systems. Their major findings are that most configuration errors result from (i) ill-formatted parameters or violations of constraints, and (ii) incompatibly of the configuration between the hardware and software environment. The first problem is not applicable to eCos and Linux because their configurators embed support for consistency preservation. However, the conflict resolution of the eCos configurators might not propose any solution—due to its incompleteness. Thus, the user may be left with an error without suggestions on how to correct it.

Bak and Ali [2] carried out a user experiment on the configuration of the Linux kernel. Their work reports four major challenges: the menu hierarchy is overly complex; feature names and description are obscure for non-experts; searching mechanisms are too primitive; and proper hardware detection is missing. Our paper converges towards similar conclusions but targets a much larger sample of users, compares results from Linux and eCos, and does not impose particular configuration scenarios on the participants. We also focus on the guidance provided by configuration tools.

7. CONCLUSION

Our survey showed that the configuration of an operating system is mostly a single user activity and that users resolve conflicting decisions as soon as they appear. Many respondents complained about the lack of guidance for making configuration decisions, and the low quality of the advice provided by the configurators. Specifically, activating an inactive option can be difficult, advice is often incomplete, and determining only the necessary options is hard. A few participants also required better support for Linux kernel updates, and one expressed the need for multiple instances of the same component in eCos.

We see two directions for future work based on this survey. First, other domains than operating systems could be explored. It would be particularly interesting to study large commercial configurators such as those used in ERPs and compare the results with the open-source projects. Secondly, both the Linux kernel and eCos configurators could be extended with better guidance mechanisms (e.g., automated support to activate inactive options).

Acknowledgements

This work is sponsored by the Interuniversity Attraction Poles Programme of the Belgian State, Belgian Science Policy, under the MoVES project.

8. REFERENCES

[1] M. Antkiewicz and K. Czarnecki. FeaturePlugin: feature modeling plug-in for Eclipse. In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange (eclipse '04)*, pages 67–72, 2004.

[2] K. Bak and K. Ali. Improving usability of the Linux kernel configuration tools. http://gsd.uwaterloo.ca/sites/default/files/cs889-report.pdf.

[3] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'07)*, pages 129–134, Limerick, Ireland, January 2007. Lero Technical Report 2007-01.

[4] T. Berger, S. She, R. Lotufo, A. Wąsowski, and K. Czarnecki. Variability modeling in the real: a

perspective from the operating systems domain. In *Proceedings of the 25th International Conference on Automated Software Engineering (ASE'10)*, pages 73–82, Antwerp, Belgium, 2010. ACM.

[5] D. Beuche. Modeling and building software product lines with pure::variants. In *Proceedings of the 2008 12th International Software Product Line Conference (SPLC '08)*, page 358, Washington, DC, USA, 2008. IEEE Computer Society.

[6] BigLever Software (Inc.). Product line engineering solutions for systems and software. http://www.biglever.com/extras/BigLever_Solution_Brochure.pdf, November 2011.

[7] L. Chen and M. Ali Babar. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, 53(4):344–362, 2011.

[8] K. Czarnecki, S. Helsen, and U. W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[9] D. Dhungana, P. Grünbacher, and R. Rabiser. DecisionKing: A flexible and extensible tool for integrated variability modeling. In *Proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'07)*, pages 119–127, Limerick, Ireland, January 2007. Lero Technical Report 2007-01.

[10] A. Hubaux, A. Classen, M. Mendonça, and P. Heymans. A preliminary review on the application of feature diagrams in practice. In *Proceedings of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10)*, pages 53–59, Linz, Austria, January 2010. Universität Duisburg-Essen.

[11] A. Hubaux, Y. Xiong, and K. Czarnecki. Configuration challenges in Linux and eCos: A survey. Technical Report GSDLAB-TR 2011-09-29, Generative Software Development Laboratory, University of Waterloo, 2011.

[12] M. Janota. Do SAT solvers make good configurators? In *Workshop on Analyses of Software Product Lines (ASPL 2008)*, pages 191–195, Limerick, Ireland, September 2008.

[13] M. Janota. *SAT Solving in Interactive Configuration*. PhD thesis, University College Dublin, 2010.

[14] K. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical report, Software Engineering Institute, Carnegie Mellon University, 1990.

[15] C. Kästner, T. Thüm, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel. FeatureIDE: A tool framework for feature-oriented software development. In *Proceedings of the 31st International Conference on Software Engineering (ICSE'09*, pages 611–614, Vancouver, Canada, 2009. IEEE.

[16] O. Koren. A study of the Linux kernel evolution. *ACM SIGOPS Operating Systems Review*, 40:110–112, 2006.

[17] M. Mendonca, M. Branco, and D. Cowan. S.P.L.O.T.: software product lines online tools. In *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA'09)*, pages 761–762, New York, NY, USA, 2009. ACM.

[18] M. Mendonca, A. Wąsowski, and K. Czarnecki. SAT-based analysis of feature models is easy. In *Proceedings of the 13th International Software Product Line Conference (SPLC'09)*, pages 231–240, San Francisco, CA, USA, 2009. Carnegie Mellon University.

[19] M. Mendonça. *Efficient Reasoning Techniques for Large Scale Feature Models*. PhD thesis, University of Waterloo, 2009.

[20] R. Michel, A. Classen, A. Hubaux, and Q. Boucher. A formal semantics for feature cardinalities in feature diagrams. In *Proceedings of the 5th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'11)*, pages 82–89, Namur, Belgium, 2011. ACM Press.

[21] A. Nöhrer and A. Egyed. Conflict resolution strategies during product configuration. In *Proceedings of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10)*, pages 107–114, Linz, Austria, 2010. Universität Duisburg-Essen.

[22] R. Rabiser, P. Grünbacher, and D. Dhungana. Requirements for product derivation support: Results from a systematic literature review and an expert survey. *Information and Software Technology*, 52(3):324 – 346, 2010.

[23] K. Schmid, R. Rabiser, and P. Grünbacher. A comparison of decision modeling approaches in product lines. In *Fifth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS'11)*, ACM International Conference Proceedings Series, pages 119–126. ACM, 2011.

[24] J. Sincero and W. Schröder-Preikschat. The Linux kernel configurator as a feature modeling tool. In *Proceedings of the 1st Workshop on Analyses of Software Product Lines (ASPL'08)*, pages 257–260, Limerick, Ireland, 2008.

[25] J. White, D. C. Schmidt, D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated diagnosis of product-line configuration errors in feature models. In *Proceedings of the 12th International Software Product Line Conference (SPLC'08)*, pages 225–234, Limercick, Ireland, 2008. IEEE Computer Society.

[26] Y. Xiong, A. Hubaux, S. She, and K. Czarnecki. Generating range fixes for software configuration. Technical Report GSDLAB-TR 2011-10-27, Generative Software Development Laboratory, University of Waterloo, 2011.

[27] Z. Yin, X. Ma, J. Zheng, Y. Zhou, L. Bairavasundaram, and S. Pasupathy. An empirical study on configuration errors in commercial and open source systems. In *Proceedings of 23rd ACM Symposium on Operating Systems Principles (SOSP)*, pages 159–172. ACM, 2011.