TABLE 5 Supported Linux Device Driver Types

Device Type	**Character**, Block, Network
Bus Type	Platform, PCI, PCMCIA, **Nobus**
Advanced Feature	**Module-parameter**, **Kernel-timer**, **Tasklet-function**, **Interrupt**, Kernel-thread, **Workqueue**

In this section, we give a case study. Several changes can be considered: change of kernel version, appending new device, and change of device driver specification. We can classify those changes into two categories: changes of code pattern and addition of new device driver specification elements.

Changes of code pattern. Changes of code pattern occur when the kernel is changed or new device is appended. Device driver specification is changed and device driver specification requires new information, then extraction code pattern needs to be changed. TABLE 6 represents an old pattern and its modified pattern. To extract name of a device, in the case of old pattern its location is the macro declaration KDSE_cdev, but in modified pattern the device name location is changed to function declaration KDSE_init.

TABLE 6 Changes of Code Pattern Configuration

Element	Old Pattern	Modified Pattern
Device / Name	static struct cdev KDSE_cdev;	static int KDSE_init(void) {…}

TABLE 7 gives an example of change of code pattern in XSL files. In the first row, old code pattern is '_cdev' in variable declaration but in the second row, modified code pattern is changed to '_init' in function declaration. In the case of the pattern '_cdev', the code pattern can only be applied to character device. The modified pattern, however, '_init' is the initializing function of a device driver, so the modified pattern can be applied to various types of devices. Simply modifying DDSER, that is, changes of code pattern can be easily supported without changes in XDDSE tool.

TABLE 7 Result of Changes of Code Pattern

```
<xsl:template match="Macro" mode="CharDev">
….
<xsl:when test="contains(child::Name,'_cdev')">
  <name> <xsl:value-of select="substring-before(.,'_cdev')"/>
  </name>
  ….
</xsl:when>
….
</xsl:template>
```

```
<xsl:template match= "FunctionDefinition"  mode="Device">
<xsl:if test="contains(child::FunctionDeclarator,'_init')">
  <name> <xsl:value-of select="substring-before
     (child::FunctionDeclarator,'_init')"/>
  </name>
  ….
</xsl:if>
….
</xsl:template>
```

Addition of new device driver specification elements. New device driver specification elements need to be added when a new device is added to Linux system. Let us consider a situation where new element of device driver specification is appended. TABLE 8 shows old element and appended element.

TABLE 8 Addition of New Specification Elements

Element	Old Element	Appended Element
Advanced /module-param	None	module-parameter: - name: param1 - name: param2 - name: param3 - name: param4 - name: param5

IV . Related Works

Source code generation and tools are studied in progress actively. For example, WinDriver [7], DriverStudio[8], Driver Development Kit [9] includes source code generation tools. WinDriver offers functions for hardware and kernel information extraction that is needed for driver frame code generation. DriverStudio offers debugging, testing and analyzing software performance tools. Windows Driver Development Kit provides a build environment, tools, driver samples, and documentation to support driver development.

V . Conclusion and Future Works

In this paper, we present a tool XDDSE for extracting device driver specification. XDDSE is designed to support its extensibility by using XSL. A case study is given for illustrating the extensibility of XDDSE.

At present, DDSE mainly covers several kinds of device and bus types. There are, however, other possible classification such as sound and video. Each usage-domain device driver has its own characteristics. They will be included in the device driver specification and the DDSER. In addition, we plan to develop source code generation tool for device drivers based on the same technology, XSL.

Reference

[1] Lee. E. A. "What's ahead for embedded software?," Computer, Volume 33, Issue 9, Sep 2000, pp. 18-26

[2] Mattias O'Nils, Johnny Oberg, Axel Jantsch, "Grammar Based Modelling and Synthesis of Device Drivers and Bus Interfaces," EUROMICRO'98, 1998, p. 10055

[3] Chikofsky.E.J, Cross.J.H "Reverse engineering and design recovery: a taxonomy," Software, IEEE, Volume 7, Issue 1, Jan 1990, pp. 13-17

[4] Laurent Reveillere, Gilles Muller, "Improving Driver Robustness : an Evaluation of the Devil Approach," Proceedings of the 2001 International Conference on Dependable Systems and Networks, 2001

[5] CDT Parser plug-in, http://www.eclipse.org/cdt/

[6] Yong Hoon Choi, Woo Il Kown, Heung Nam Kim, "Code generation for Linux device driver," 2006. ICACT 2006, pp. 4

[7] WinDriver(http://www.jungo.com/windriver_usb_pci_driver_development_software.html

[8] DriverStudio(compuware) http://www.compuware.co.kr/products/driverstudio/ds/

[9] Windows Driver Development Kit http://www.microsoft.com/whdc/devtools/ddk