20.03.2015	**Advanced Software Engineering** **University of Vienna** **Forschungsgruppe Software Architecture**		
Kennzahl	*Matrikelnummer*	*FAMILIENNAME*	*Vorname*

8	12	10	12	4	14		60
1	2	3	4	5	6		

Bitte lesen Sie diesen Abschnitt. Sie haben **60** Minuten Zeit die Fragen zu beantworten.

Hinweise:

- Füllen Sie Kennzahl, Matrikelnummer, Familienname und Vorname zuerst aus.
- Während der Prüfung sind keinerlei Unterlagen erlaubt!
- Technische Hilfsmittel wie Übersetzungscomputer, Taschenrechner, Mobiltelefone, etc. sind nicht erlaubt!
- Wenn Sie Probleme beim Verstehen einer Frage haben, fragen Sie.
- Sie können die Fragen sowohl auf Deutsch als auch auf Englisch beantworten.
- Schreiben Sie mit dokumentenechten Schreibutensilien (kein Bleistift).
- Sie können den Appendix (letzte Seite) von der Prüfung abtrennen.

- Viel Erfolg!

Task 1 / Aufgabe 1 (8 Points / Punkte)

What are the criteria that have to be considered when choosing between a scripting language and a system programming language for a task at hand? Explain for each criterion why it will cause the use of a scripting language or a system programming language.

Welche Kriterien sind zu beachten, wenn Sie sich zur Entwicklung eines Programmes zwischen einer Skriptsprache und einer Systemprogrammiersprache entscheiden müssen? Erklären Sie für jedes Kriterium, warum dies zum Einsatz einer Skriptsprache oder einer Systemprogrammiersprache führt.

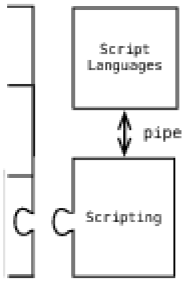
Task 2 / Aufgabe 2 (12 Points / Punkte)

Audacity is a popular open source sound recorder and audio editor that runs on several platforms. Document the architectural design decision that is described in the text below using the (simplified) AD template from Tyree / Ackermann.

Audacity ist ein beliebtes Open Source System zum Aufnehmen und Bearbeiten von Audiodateien, das für verschiedene Plattformen verfügbar ist.
Dokumentieren Sie die Architectural Decision, die im Text unterhalb beschrieben wurde, mittels der (vereinfachten) Vorlage zur Dokumentation von Ads von Tyree / Ackermann.

Audacity has an experimental plugin that supports multiple scripting languages. It provides a scripting interface over a named pipe. The commands exposed via scripting are in a textual format, as are the responses. As long as the user's scripting language can write text to and read text from a named pipe, the scripting language can drive Audacity. Audio and other high-volume data does not need to travel on the pipe (Figure 2.6).

Figure 2.6



The plugin itself knows nothing about the content of the text traffic that it carries. It is only responsible for conveying it. The plugin interface (or rudimentary extension point) used by the scripting plugin to plug in to Audacity already exposes Audacity commands in textual format. So, the scripting plugin is small, its main content being code for the pipe.

Unfortunately a pipe introduces similar security risks to having a TCP/IP connection—and we've ruled out TCP/IP connections for Audacity on security grounds. To reduce that risk the plugin is an optional DLL. You have to make a deliberate decision to obtain and use it and it comes with a health/security warning.

After the scripting feature had already been started, a suggestion surfaced in the feature requests page of our wiki that we should consider using KDE's D-Bus standard to provide an inter-process call mechanism using TCP/IP. We'd already started going down a different route but it still might make sense to adapt the interface we've ended up with to support D-Bus.

Origins of Scripting Code

The scripting feature grew from an enthusiast's adaptation of Audacity for a particular need that was heading in the direction of being a fork. These features, together called CleanSpeech, provide for mp3 conversion of sermons. CleanSpeech adds new effects such as truncate silence—the effect finds and cuts out long silences in audio—and the ability to apply a fixed sequence of existing noise removal effects, normalization and mp3 conversion to a whole batch of audio recordings. We wanted some of the excellent functionality in this, but the way it was written was too special case for Audacity. Bringing it into mainstream Audacity led us to code for a flexible sequence rather than a fixed sequence. The flexible sequence could use any of the effects via a look-up table for command names and a Shuttle class to persist the command parameters to a textual format in user preferences. This feature is called *batch chains*. Very deliberately we stopped short of adding conditionals or calculation to avoid inventing an ad hoc scripting language.

In retrospect the effort to avoid a fork has been well worthwhile. There is still a CleanSpeech mode buried in Audacity that can be set by modifying a preference. It also cuts down the user interface, removing advanced features. A simplified version of Audacity has been requested for other uses, most notably in schools. The problem is that each person's view of which are the advanced features and which are the essential ones is different. We've subsequently implemented a simple hack that leverages the translation mechanism. When the translation of a menu item starts with a "#" it is no longer shown in the menus. That way people who want to reduce the menus can make choices themselves without recompiling—more general and less invasive than the mCleanSpeech flag in Audacity, which in time we may be able to remove entirely.

The CleanSpeech work gave us batch chains and the ability to truncate silence. Both have attracted additional improvement from outside the core team. Batch chains directly led on to the scripting feature. That in turn has begun the process of supporting more general purpose plugins to adapt Audacity.

AD shortname		AD name	

Problem Statement			
Decision drivers			
Alternatives			
Recommendation			
Decision Outcomes			
Decision Outcomes			
	Status		
	Chosen alternative		
	Justification		
	Consequences		
	Assumptions		

Task 3 / Aufgabe 3 (10 Points / Punkte)

In the context of model-driven development, which statements are true?

- □ The difference between “generated models” vs. “models are part of the software and interpreted or evaluated at runtime” is similar to the difference between “compiler” and “interpreter”.
- □ The variant “models are part of the software and interpreted or evaluated at runtime” usually leads to faster execution times than using “generated models”.
- □ The variant “models are part of the software and interpreted or evaluated at runtime” supports changes of the models at runtime, which is usually not the case for “generated models”.

Explain for each statement, why it is true or false!

Im Kontext von modell-getriebener Entwicklung: Welche der folgenden Aussagen ist wahr?

- □ Der Unterschied zwischen “generierten Modellen” vs. “Modelle sind Teil der Software und werden zur Laufzeit interpretiert oder evaluiert” ist vergleichbar mit dem Unterschied zwischen “Übersetzer” und “Interpreter”.
- □ Die Variante „Modelle sind Teil der Software und werden zur Laufzeit interpretiert oder evaluiert” führt üblicherweise zu schnelleren Ausführungszeiten als bei „generierten Modellen“.
- □ Die Variante „Modelle sind Teil der Software und werden zur Laufzeit interpretiert oder evaluiert” erlaubt Änderungen an den Modellen zur Laufzeit, während dies bei „generierten Modellen“ normalerweise nicht der Fall ist.

Begründen Sie für jede Aussagen, warum diese wahr oder falsch ist!

Task 4 / Aufgabe 4 (12 Points / Punkte)

a) **Explain** the term Architectural / Design Analysis! **Name** and **explain** the goals of architectural analysis!
b) **Name** and **explain** the techniques for performing architectural analysis!

a) **Erklären** Sie den Begriff "Architekur- / Designanalyse"**. Nennen** und **erklären** Sie die Ziele der Architektur-/ Designanalyse.
b) **Nennen** und **erklären** Sie die Techniken, welche für Designanalyse eingesetzt werden können.

Task 5 / Aufgabe 5 (4 Points / Punkte)

Name at least 4 different means to define / describe EMF meta models.

Nenn Sie mindestens 4 Möglichkeiten um ein EMF Metamodell zu definieren beziehungsweise zu beschreiben?

Task 6 / Aufgabe 6 (14 Points / Punkte)

Create a valid Xtext grammar for the Ecore diagram shown in the appendix (on the next page).

Erstellen Sie eine gültige Xtext-Grammatik für das Ecore-Diagramm im Anhang (auf der nächsten Seite).

Appendix

Task 6

XText Grammar