26.01.2015	**Advanced Software Engineering** **University of Vienna** **Forschungsgruppe Software Architecture**		
Kennzahl	*Matrikelnummer*	*FAMILIENNAME*	*Vorname*

6	12	12	12	12	6			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 (6 Points / Punkte)

a) Explain the term “model” in general!
b) Which models are used in Software Development? Give at least 2 examples for models used in Software Development and describe how they are used.

a) Erklären Sie den Begriff „Modell“ im Allgemeinen.
b) Welche Modelle werden in der Softwareentwicklung eingesetzt? Nennen und erklären Sie 2 Beispiele für Modelle, die in der Softwareentwicklung verwendet werden.

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.

One of the challenges faced by Audacity is supporting insertions and deletions into audio recordings that may be hours long. Recordings can easily be too long to fit in available RAM. If an audio recording is in a single disk file, inserting audio somewhere near the start of that file could mean moving a lot of data to make way. Copying that data on disk would be time consuming and mean that Audacity could then not respond rapidly to simple edits.

Audacity's solution to this is to divide audio files into many BlockFiles, each of which could be around 1 MB. This is the main reason Audacity has its own audio file format, a master file with the extension .aup. It is an XML file which coordinates the various blocks. Changes near the start of a long audio recording might affect just one block and the master .aup file.

BlockFiles balance two conflicting forces. We can insert and delete audio without excessive copying, and during playback we are guaranteed to get reasonably large chunks of audio with each request to the disk. The smaller the blocks, the more potential disk requests to fetch the same amount of audio data; the larger the blocks, the more copying on insertions and deletions.

Audacity's BlockFiles never have internal free space and they never grow beyond the maximum block size. To keep this true when we insert or delete we may end up copying up to one block's worth of data. When we don't need a BlockFile anymore we delete it. The BlockFiles are reference counted so if we delete some audio, the relevant BlockFiles will still hang around to support the undo mechanism until we save. There is never a need to garbage collect free space within Audacity BlockFiles, which we would need to do with an all-in-one-file approach.

A refinement in the BlockFile system is that the blocks needn't be files created by Audacity. They can be references to subsections of audio files such as a timespan from audio stored in the .wav format. A user can create an Audacity project, import audio from a .wav file and mix a number of tracks whilst only creating BlockFiles for the summary information. This saves disk space and saves time in copying audio. All told it is, however, a rather bad idea. Far too many of our users have removed the original audio .wav file thinking there will be a complete copy in the Audacity project folder. That's not so and without the original .wav file the audio project can no longer be played. The default in Audacity nowadays is to always copy imported audio, creating new BlockFiles in the process.

The BlockFile solution ran into problems on Windows systems where having a large number of BlockFiles performed very poorly. This appeared to be because Windows was much slower handling files when there were many in the same directory, a similar problem to the slowdown with large numbers of widgets. A later addition was made to use a hierarchy of subdirectories, never with more than a hundred files in each subdirectory.

The main problem with the BlockFile structure is that it is exposed to end users. We often hear from users who move the .aup file and don't realize they also need to move the folder containing all the BlockFiles too. It would be better if Audacity projects were a single file with Audacity taking responsibility for how the space inside the file is used. If anything this would increase performance rather than reduce it. The main additional code needed would be for garbage collection. A simple approach to that would be to copy the blocks to a new file when saving if more than a set percentage of the file were unused.

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

Task 3 / Aufgabe 3 (12 Points / Punkte)

a) What is software architecture evolution and why is it relevant?
b) Name and explain the problems in software architecture evolution!
c) How can the problems in software architecture evolution be tackled?

a) Erklären Sie den Begriff "Software Architecture Evolution"! Warum ist Software Architecture Evolution wichtig?
b) Nennen und beschreiben Sie die Probleme bei der Evolution von Software Architekturen auftreten!
c) Wie können die bei der Software Architektur Evolution auftretenden Probleme bekämpft werden?

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 (12 Points / Punkte)

Create a valid instance of the XText grammar for the domain specific language that is described below. Use each grammar rule at least once.
For your convenience, you will find the XText grammar at the Appendix as well, which can be separated from this exam.

Erstellen Sie eine Instanz der folgenden Grammatik einer domänenspezifischen Sprache.
Verwenden Sie jede Grammatikregel mindestens einmal.
Zur leichteren Handhabung finden Sie die Code-Fragmente im Anhang, den Sie von dieser Prüfung abtrennen können.

```
FastFoodDomain:
	{FastFoodDomain}
	"FastFoodDomain" "{"
	(("Restaurants" "{" restaurants+=Restaurant* "}")?
	& ("Ingredients" "{" ingredients+=Ingredient* "}")?)
	"}";
Restaurant: FastFoodRestaurant | TakeAwayRestaurant;
FastFoodRestaurant:
	"FastFoodRestaurant" name=ID "{"
	(("cash:" cash=EFloat)?
	& "number of tables: " numberOfTables=EInt
	& ("Products" "{" producs+=Product+ "}")
	& ("Orders" "{" orders+=Order+ "}"))
	"}";
TakeAwayRestaurant:
	"TakeAwayRestaurant" name=ID "{"
	("cash:" cash=EFloat)?
	& ("Products" "{" producs+=Product+ "}")?
	& ("Orders" "{" orders+=Order* "}")
	"}";
Order:
	"Order" "{"
	"Items" "{" items+=OrderItem* "}"
	"date:" date=EDate
	"}";
OrderItem:
	"OrderItem" product=[Product] "{"
	"amount:" amount=EInt
	"price:" price=EFloat
	"}";
Product:
	"Product" name=ID "{"
	(("Products" "{" products+=[Product]* "}")?
	& ("ProductIngredients" "{" productIngredients+=ProductIngredient* "}")?
	& ("price:" price=EFloat)?)
	"}";
ProductIngredient:
	"ProductIngredient" ingredient=[Ingredient] "{"
	(("amount:" amount=EFloat)?
	& ("step:" step=EInt)?
	& ("instruction:" preparationInstruction=EString)?)
	"}";
Ingredient: "Ingredient" name=ID ";";

EFloat returns ecore::EFloat: '-'? INT? '.' INT (('E' | 'e') '-'? INT)?;
EInt returns ecore::EInt: '-'? INT;
EString returns ecore::EString: STRING | ID;
Entity: "timestamp" '=' EDate;
EDate returns ecore::EDate: EString;
```

Task 6 / Aufgabe 6 (6 Points / Punkte)

Name and **explain** 3 characteristics of scripting languages!

Nennen und **beschreiben** Sie 3 Charakteristika von Skriptsprachen!

Appendix

Task 5

XText Grammar

```
FastFoodDomain:
	{FastFoodDomain}
	"FastFoodDomain" "{"
	(("Restaurants" "{" restaurants+=Restaurant* "}")?
	& ("Ingredients" "{" ingredients+=Ingredient* "}")?)
	"}";
Restaurant: FastFoodRestaurant | TakeAwayRestaurant;
FastFoodRestaurant:
	"FastFoodRestaurant" name=ID "{"
	(("cash:" cash=EFloat)?
	& "number of tables: " numberOfTables=EInt
	& ("Products" "{" producs+=Product+ "}")
	& ("Orders" "{" orders+=Order+ "}"))
	"}";
TakeAwayRestaurant:
	"TakeAwayRestaurant" name=ID "{"
	("cash:" cash=EFloat)?
	& ("Products" "{" producs+=Product+ "}")?
	& ("Orders" "{" orders+=Order* "}")
	"}";
Order:
	"Order" "{"
	"Items" "{" items+=OrderItem* "}"
	"date:" date=EDate
	"}";
OrderItem:
	"OrderItem" product=[Product] "{"
	"amount:" amount=EInt
	"price:" price=EFloat
	"}";
Product:
	"Product" name=ID "{"
	(("Products" "{" products+=[Product]* "}")?
	& ("ProductIngredients" "{" productIngredients+=ProductIngredient* "}")?
	& ("price:" price=EFloat)?)
	"}";
ProductIngredient:
	"ProductIngredient" ingredient=[Ingredient] "{"
	(("amount:" amount=EFloat)?
	& ("step:" step=EInt)?
	& ("instruction:" preparationInstruction=EString)?)
	"}";
Ingredient: "Ingredient" name=ID ";";

EFloat returns ecore::EFloat: '-'? INT? '.' INT (('E' | 'e') '-'? INT)?;
EInt returns ecore::EInt: '-'? INT;
EString returns ecore::EString: STRING | ID;
Entity: "timestamp" '=' EDate;
EDate returns ecore::EDate: EString;
```