05.06.2015	**Advanced Software Engineering** **University of Vienna** **Forschungsgruppe Software Architecture**		
Kennzahl	*Matrikelnummer*	*FAMILIENNAME*	*Vorname*

8	12	10	8	4	4	14		60
1	2	3	4	5	6	7		

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!

Please read this section. You have **60** minutes, in which to answer the questions.

Tips:

- Before you start, please fill in your name, surname and student ID number.
- No teaching material is permitted during the test!
- Technical aids such as translators, pocket calculators, mobile telephones, etc., are not allowed!
- If you do not understand a question, please ask the supervisors for explanations.
- You can answer the questions in German as well as in English.
- Please write using indelible writing utensils (no pencil) .
- You can rip off the appendix (last page) from the test document for easier use.

- Good luck!

Task 1 / Aufgabe 1 (8 Points / Punkte)

a) What is a Domain-Specific Language (DSL)? Explain its differences to general purpose languages, and provide two examples of DSLs used in the real world.
b) Which of the following statements are true? For statements that are not true, explain why they are not true:

- One DSL can have many concrete syntaxes, but has one meta-model (abstract syntax)
- The semantics of the DSL must always be described precisely in a formal language
- In Xtext, source code is directly generated from a grammar.
- Model-driven design is suited even for one-off problems, as it imposes little overhead effort.

a) Was ist eine Domain-Specific Language (DSL)? Erklären Sie den Unterschied zu General Purpose Languages und geben Sie zwei Beispiele von DSLs, die in der Praxis angewandt werden.
b) Welche der folgenden Aussagen sind wahr? Erklären Sie für falsche Aussagen, warum diese falsch sind:

- Eine DSL kann mehrere konkrete Syntaxe, aber nur ein Meta-Modell (abstrakte Syntax) haben.
- Die Semantik einer DSL muss immer in einer formalen Sprache genau beschriben sein.
- In Xtext wird Quellcode direkt aus der Grammatik generiert.
- Model-Driven Design ist auch für Einzelanwendungen geeignet, da es sehr wenig zusätzlichen Aufwand erfordert.

Task 2 / Aufgabe 2 (12 Points / Punkte)

nginx (pronounced "engine x") is a widely used free open source web server.
Document the architectural design decision that is described in the text below using the (simplified) AD template from Tyree / Ackermann.

nginx (ausgesprochen „engine x“) ist ein populärer kostenloser Open-Source Webserver.
Dokumentieren Sie die Architectural Decision, die im Text unterhalb beschrieben wurde, mittels der (vereinfachten) Vorlage zur Dokumentation von Ads von Tyree / Ackermann.

14.2 Overview of nginx Architecture

Traditional process- or thread-based models of handling concurrent connections involve handling each connection with a separate process or thread, and blocking on network or input/output operations. Depending on the application, it can be very inefficient in terms of memory and CPU consumption. Spawning a separate process or thread requires preparation of a new runtime environment, including allocation of heap and stack memory, and the creation of a new execution context. Additional CPU time is also spent creating these items, which can eventually lead to poor performance due to thread thrashing on excessive context switching. All of these complications manifest themselves in older web server architectures like Apache's. This is a tradeoff between offering a rich set of generally applicable features and optimized usage of server resources.

From the very beginning, nginx was meant to be a specialized tool to achieve more performance, density and economical use of server resources while enabling dynamic growth of a website, so it has followed a different model. It was actually inspired by the ongoing development of advanced event-based mechanisms in a variety of operating systems. What resulted is a modular, event-driven, asynchronous, single-threaded, non-blocking architecture which became the foundation of nginx code.

nginx uses multiplexing and event notifications heavily, and dedicates specific tasks to separate processes. Connections are processed in a highly efficient run-loop in a limited number of single-threaded processes called workers. Within each worker nginx can handle many thousands of concurrent connections and requests per second.

Code Structure
The nginx worker code includes the core and the functional modules. The core of nginx is responsible for maintaining a tight run-loop and executing appropriate sections of modules' code on each stage of request processing. Modules constitute most of the presentation and application layer functionality. Modules read from and write to the network and storage, transform content, do outbound filtering, apply server-side include actions and pass the requests to the upstream servers when proxying is activated.

nginx's modular architecture generally allows developers to extend the set of web server features without modifying the nginx core. nginx modules come in slightly different incarnations, namely core modules, event modules, phase handlers, protocols, variable handlers, filters, upstreams and load balancers. At this time, nginx doesn't support dynamically loaded modules; i.e., modules are compiled along with the core at build stage. However, support for loadable modules and ABI is planned for the future major releases.

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

Task 3 / Aufgabe 3 (8 Points / Punkte)

What does "Ousterhout's Dichotomy" say about the relation of scripting and system programming languages? Explain Ousterhout’s claim in detail and answer the following questions:

- What does "gluing" mean?
- Why is "typing" un-/important?

Was besagt "Ousterhouts Dichotomie" über das Verhältnis von Skript- und Systemprogrammiersprachen? Erläutern Sie Ousterhouts Behauptung im Detail und beantworten Sie die folgenden Fragen:

- Was bedeutet "gluing"?
- Warum ist "typing" un-/wichtig?

Task 4 / Aufgabe 4 (10 Points / Punkte)

a) Explain the term "software architecture evolution"! Why is it relevant?
b) Name and explain the problems that occur in software architecture evolution!
c) How can the software architecture evolution problems 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 Architecture Evolution auftretenden Probleme bekämpft werden?

Task 5 / Aufgabe 5 (4 Points / Punkte)

Name at least 4 different ways to define / describe EMF metamodels.

Nenn Sie mindestens 4 Möglichkeiten, ein EMF Metamodell zu definieren bzw. zu beschreiben.

Task 6 / Aufgabe 6 (4 Points / Punkte)

Which are the architectural principles that must be followed when designing a "good" software architecture? Explain them briefly in your own words.

Nennen und erklären Sie die Prinzipien, denen gefolgt werden sollte, um eine “gute” Softwarearchitektur zu gewährleisten.

Task 7 / Aufgabe 7 (14 Points / Punkte)

a) Create a valid Xtext grammar for the music library described below:

- Every artist has a unique name and has produced a number of music pieces.
- Music pieces are published either in albums or in singles, are vocal or instrumental, and are performed by one or more artists.
- Each album or single has a unique name and contains an ordered list of music pieces, the performing artist(s), the recording company, the month and the year of its publication, and a genre. Available genres are "Rock", "Pop", "Jazz", "Folk", and "Classical".
- A month should be written as digits and should only allow the values between one and twelve. A year should be written as four digits, limited between the years 1900 and 2100.
- A playlist consists of individual songs or entire albums/singles in custom order.

b) Create an instance of your grammar, that uses each rule at least once.

a) Erstellen Sie eine Xtext Grammatik für die unten beschriebene Musiksammlung:

- Jeder Künstler hat einen eindeutigen Namen und hat beliebig viele Musikstücke produziert.
- Musikstücke sind in Alben oder Singles veröffentlicht, sind entweder vokal oder instrumental, und werden von einem oder mehreren Künstlern gespielt.
- Jedes Album oder Single hat einen eindeutigen Namen, und enthält eine geordnete Liste der enthaltenen Musikstücke, die mitwirkenden Künstler, den Musiklabel, den Monat sowie das Jahr der Veröffentlichung und eins der folgenden Genres: "Rock", "Pop", "Jazz", "Folk" und "Classical".
- Der Monat der Veröffentlichung sollte nur Ziffern zwischen eins und zwölf als Eingabe erlauben. Das Jahr der Veröffentlichung sollte auf die Jahre zwischen 1900 und 2100 beschränkt sein.
- Eine Wiedergabeliste (Playlist) enthält mehrere Lieder oder gleich ganze Albums oder Singles in beliebiger Reihenfolge.

b) Erstellen Sie eine Instanz Ihrer Grammatik, wobei jede Regel mindestens einmal genutzt werden muss.