Curriculum für

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Modul
DSL

Domänenspezifische Sprachen

2023.1-rev0-DE-20230508

Inhaltsverzeichnis

© (Copyright), International Software Architecture Qualification Board e. V. (iSAQB® e. V.) 2023

Die Nutzung des Lehrplans ist nur unter den nachfolgenden Voraussetzungen erlaubt:

1. Sie möchten das Zertifikat zum CPSA Certified Professional for Software Architecture Foundation Level® oder CPSA Certified Professional for Software Architecture Advanced Level® erwerben. Für den Erwerb des Zertifikats ist es gestattet, die Text-Dokumente und/oder Lehrpläne zu nutzen, indem eine Arbeitskopie für den eigenen Rechner erstellt wird. Soll eine darüber hinausgehende Nutzung der Dokumente und/oder Lehrpläne erfolgen, zum Beispiel zur Weiterverbreitung an Dritte, Werbung etc., bitte unter info@isaqb.org nachfragen. Es müsste dann ein eigener Lizenzvertrag geschlossen werden.
2. Sind Sie Trainer oder Trainingsprovider, ist die Nutzung der Dokumente und/oder Lehrpläne nach Erwerb einer Nutzungslizenz möglich. Hierzu bitte unter info@isaqb.org nachfragen. Lizenzverträge, die alles umfassend regeln, sind vorhanden.
3. Falls Sie weder unter die Kategorie 1. noch unter die Kategorie 2. fallen, aber dennoch die Dokumente und/oder Lehrpläne nutzen möchten, nehmen Sie bitte ebenfalls Kontakt unter info@isaqb.org zum iSAQB e. V. auf. Sie werden dort über die Möglichkeit des Erwerbs entsprechender Lizenzen im Rahmen der vorhandenen Lizenzverträge informiert und können die gewünschten Nutzungsgenehmigungen erhalten.

Wichtiger Hinweis

Grundsätzlich weisen wir darauf hin, dass dieser Lehrplan urheberrechtlich geschützt ist. Alle Rechte an diesen Copyrights stehen ausschließlich dem International Software Architecture Qualification Board e. V. (iSAQB® e. V.) zu.

Die Abkürzung "e. V." ist Teil des offiziellen Namens des iSAQB und steht für "eingetragener Verein", der seinen Status als juristische Person nach deutschem Recht beschreibt. Der Einfachheit halber wird iSAQB e. V. im Folgenden ohne die Verwendung dieser Abkürzung als iSAQB bezeichnet.

Verzeichnis der Lernziele

Einführung: Allgemeines zum iSAQB Advanced Level

Was vermittelt ein Advanced Level Modul?

Das Modul kann unabhängig von einer CPSA-F-Zertifizierung besucht werden.

- Der iSAQB Advanced Level bietet eine modulare Ausbildung in drei Kompetenzbereichen mit flexibel gestaltbaren Ausbildungswegen. Er berücksichtigt individuelle Neigungen und Schwerpunkte.
- Die Zertifizierung erfolgt als Hausarbeit. Die Bewertung und mündliche Prüfung wird durch vom iSAQB benannte Expert:innen vorgenommen.

Was können Absolventen des Advanced Level (CPSA-A)?

CPSA-A-Absolventen können:

- eigenständig und methodisch fundiert mittlere bis große IT-Systeme entwerfen
- in IT-Systemen mittlerer bis hoher Kritikalität technische und inhaltliche Verantwortung übernehmen
- Maßnahmen zur Erreichung von Qualitätsanforderungen konzeptionieren, entwerfen und dokumentieren sowie Entwicklungsteams bei der Umsetzung dieser Maßnahmen begleiten
- architekturrelevante Kommunikation in mittleren bis großen Entwicklungsteams steuern und durchführen

Voraussetzungen zur CPSA-A-Zertifizierung

- erfolgreiche Ausbildung und Zertifizierung zum Certified Professional for Software Architecture, Foundation Level® (CPSA-F)
- mindestens drei Jahre Vollzeit-Berufserfahrung in der IT-Branche; dabei Mitarbeit an Entwurf und Entwicklung von mindestens zwei unterschiedlichen IT-Systemen
    - Ausnahmen sind auf Antrag zulässig (etwa: Mitarbeit in Open-Source-Projekten)
- Aus- und Weiterbildung im Rahmen von iSAQB-Advanced-Level-Schulungen im Umfang von mindestens 70 Credit Points aus mindestens drei unterschiedlichen Kompetenzbereichen
- erfolgreiche Bearbeitung der CPSA-A-Zertifizierungsprüfung

Grundlegendes

Was vermittelt das Modul „DSL“?

Viele Domänen enthalten komplexe Regelwerke, Abläufe oder Beschreibungen von Domänenentitäten. Nicht immer ist die primäre Implementierungssprache eines Projekts auch die richtige Sprache, um diese Aspekte auszudrücken, zum Beispiel wenn

- diese Aspekte von den Benutzer:innen konfiguriert werden können,
- die Implementierungssprache zu ausdrucksstark ist, um Eigenschaften Sicherheit, Laufzeit oder Termination statisch zu garantieren,
- die Implementierungssprache zu ausdrucksschwach ist, um diese Aspekte lesbar und nachvollziehbar zu beschreiben.

In solchen Projekten explizit eine eigene Sprache für diese Aspekte zu definieren kann helfen, die dadurch entstehende Komplexität zu bändigen. Es entsteht eine *domänenspezifische Sprache* oder *DSL* (für *domain-specific language*). Gut gemachte DSLs:

- tragen in hohem Maße zum *separation of concerns* bei, indem Beschreibung von Implementierung entkoppelt wird,
- befähigen Benutzer:innen, selbständig mit der Software Aufgaben zu lösen, für die sonst Entwickler:innen die Software erweitern müssten,
- ermöglichen Low-Code-Ansätze, bei denen Aufgaben mit weniger Code gelöst werden können, als es in der Implementierungssprache des Projekts möglich wäre, und
- verbessern eine ganzen Reihe architektonischer Qualitätsaspekte, darunter Adaptierbarkeit, Modifizierbarkeit, Analysierbarkeit und Sicherheit.

Für Design und Implementierung einer DSL gibt es einen großen Korpus an Techniken und Erfahrung aus dem Programmiersprachen-Design und dem Compilerbau. Dieses Modul macht Architekt:innen mit den wichtigsten Elementen dieses Korpus bekannt. Dieser versetzt sie in die Lage:

- in der Architektur sinnvolle Stellen für DSLs zu finden,
- nützliche und benutzerfreundliche DSLs systematisch zu entwickeln und
- DSLs als integralen Aspekt des Architektur-Designs anzuwenden.

Struktur des Lehrplans und empfohlene zeitliche Aufteilung

Inhalt	Empfohlene Mindestdauer (min)
1. Thema mit Einleitung	180
2. Thema über xz	150
3. Thema mit viel Theorie	120
4. Thema mit xy und Beispiel	180
5. Thema mit abc und d	210
6. Thema mit Abschlussbeispiel	120
Summe	960 (16h)

Dauer, Didaktik und weitere Details

Die unten genannten Zeiten sind Empfehlungen. Die Dauer einer Schulung zum Modul DSL sollte mindestens **3** Tage betragen, kann aber länger sein. Anbieter können sich durch Dauer, Didaktik, Art und Aufbau der Übungen sowie der detaillierten Kursgliederung voneinander unterscheiden. Insbesondere die Art der Beispiele und Übungen lässt der Lehrplan komplett offen.

Lizenzierte Schulungen zu DSL tragen zur Zulassung zur abschließenden Advanced-Level-Zertifizierungsprüfung folgende Credit Points) bei:

Methodische Kompetenz:	**10** Punkte
Technische Kompetenz:	**10** Punkte
Kommunikative Kompetenz:	**10** Punkte

Gliederung des Lehrplans

Die einzelnen Abschnitte des Lehrplans sind gemäß folgender Gliederung beschrieben:

- **Begriffe/Konzepte**: Wesentliche Kernbegriffe dieses Themas.
- **Unterrichts-/Übungszeit**: Legt die Unterrichts- und Übungszeit fest, die für dieses Thema bzw. dessen Übung in einer akkreditierten Schulung mindestens aufgewendet werden muss.
- **Lernziele**: Beschreibt die zu vermittelnden Inhalte inklusive ihrer Kernbegriffe und -konzepte.

Dieser Abschnitt skizziert damit auch die zu erwerbenden Kenntnisse in entsprechenden Schulungen.

Ergänzende Informationen, Begriffe, Übersetzungen

Soweit für das Verständnis des Lehrplans erforderlich, haben wir Fachbegriffe ins iSAQB-Glossar aufgenommen, definiert und bei Bedarf durch die Übersetzungen der Originalliteratur ergänzt.

1. Einführung und Motivation

Dauer: 60 Min.	Übungszeit: 0 Min.

1.1. Begriffe und Konzepte

Begriff 1, Begriff 2, Begriff 3

1.2. Lernziele

1.3. Referenzen

[Aho et al. 2006], [Evans 2003], [Nystrom 2021], [Wąsowski and Berger 2023], [Ghosh 2010]

2. Syntax

Dauer: 120 Min.	Übungszeit: 60 Min.

2.1. Begriffe und Konzepte

Begriff 1, Begriff 2, Begriff 3

2.2. Lernziele

2.3. Referenzen

[Wilhelm et al. 2021], [Leermakers 1993], [Culpepper et al. 2019], [Völter et al. 2014], [Wąsowski and Berger 2023], [Ghosh 2010]

3. Semantik

Dauer: 180 Min.	Übungszeit: 120 Min.

3.1. Begriffe und Konzepte

Begriff 1, Begriff 2, Begriff 3

3.2. Lernziele

3.3. Referenzen

[Nipkow and Klein 2014], [Baader and Nipkow 1998]

4. Sprachdesign

Dauer: 180 Min.	Übungszeit: 120 Min.

4.1. Begriffe und Konzepte

Begriff 1, Begriff 2, Begriff 3

4.2. Lernziele

4.3. Referenzen

[Pierce 2002], [Peyton Jones and Eber 2003], [Elliott and Hudak 1997], [Pretnar 2015], [Alama 2020], [Wąsowski and Berger 2023], [Ghosh 2010]

5. Werkzeuge

Dauer: 120 Min.	Übungszeit: 60 Min.

5.1. Begriffe und Konzepte

Begriff 1, Begriff 2, Begriff 3

5.2. Lernziele

5.3. Referenzen

[Bettini 2016], [Steinberg et al. 2008], [Völter 2013], [Felleisen et al. 2009], [Erdweg at el. 2015], [Humer et al. 2014], [Parr 2010], [Kleppe 2008]

6. Beispiele

Dauer: 60 Min.	Übungszeit: 0 Min.

Dieser Abschnitt ist nicht prüfungsrelevant.

6.1. Begriffe und Konzepte

Innerhalb jeder lizenzierten Schulung muss mindestens ein Beispiel für DSL vorgestellt werden.

Art und Ausprägung der vorgestellten Beispiele können von der Schulung bzw. den Interessen der Teilnehmer abhängen und werden seitens iSAQB nicht vorgegeben.

Referenzen

Dieser Abschnitt enthält Quellenangaben, die ganz oder teilweise im Curriculum referenziert werden.

A

- [Aho et al. 2006] Aho, Alfred, Lam, Monica, Sethi, Ravi, Ullman, Jeffrey: Compilers: Principles, Techniques, and Tools. Addison Wesley, 2006.
- [Alama 2020] Alama, Jesse: Language-Oriented Programming in Racket - A Cultural Anthropology. Gumroad, 2020. https://jessealama.gumroad.com/l/lop-in-racket-cultural-anthro

B

- [Baader and Nipkow 1998] Baader, Franz, Nipkow, Tobias: Term Rewriting and All That. Cambridge University Press, 1998.
- [Bettini 2016] Bettini, Lorenzo: Implementing Domain-Specific Languages with Xtext and Xtend - Second Edition. Packt, 2016.

C

- [Culpepper et al. 2019] Culpepper, Ryan, Felleisen, Matthias, Flatt, Matthew, Krishnamurthi, Shriram: From Macros to DSLs: The Evolution of Racket. Summit on Advances in Programming Languages, 2019. https://cs.brown.edu/~sk/Publications/Papers/Published/cffk-macros-to-dsls/

E

- [Elliott and Hudak 1997] Elliott, Conal, Hudak, Paul: Functional reactive animation. In ACM SIGPLAN International Conference on Functional Programming (ICFP '97). pages 263-273. ACM, Amsterdam.
- [Evans 2003] Evans, Eric J.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003.
- [Erdweg at el. 2015] Sebastian, Erdweg et al.: Evaluating and comparing language workbenches: Existing results and benchmarks for the future. In Computer Languages, Systems & Structures, 44:24–47, 2015.

F

- [Felleisen et al. 2009] Felleisen, Matthias, Finder, Robert Bruce, Flatt, Matthew: Semantics Engineering with PLT Redex. MIT Press, 2009.

G

- [Ghosh 2010] Ghosh Debasish: DSLs in Action. Manning, 2010.

H

- [Humer et al. 2014] Humer, Christian, et al.: A Domain-Specific Language for Building Self-Optimizing AST Interpreters. In Proceedings of the International Conference on Generative Programming: Concepts and Experiences (GPCE'14), 2014.

K

- [Kleppe 2008] Kleppe, Anneke: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels . Addison-Wesley, 2008.

L

- [Leermakers 1993] Leermakers, René: The Functional Treatment of Parsing. Springer Science, 1993.

M

- [Metsker 2001] Metsker, Steven John: Building Parsers With Java. Addison Wesley, 2001.

N

- [Nipkow and Klein 2014] Nipkow, Tobias, Klein, Gerwin: Concrete Semantics. Springer, 2014.
- [Nystrom 2021] Nystrom, Robert: Crafting Interpreters. Genever Benning, 2021.

P

- [Parr 2010] Parr, Terrence: Language Implementation Patterns. O'Reilly, 2010.
- [Pierce 2002] Pierce, Benjamin C.: Types and Programming Languages. MIT Press, 2002.
- [Peyton Jones and Eber 2003] Peyton Jones, Simon, Eber, Jean-Marc: How to write a financial contract. Chapter in "The Fun of Programming", ed. Gibbons and de Moor, Palgrave Macmillan 2003.
- [Pretnar 2015] Matija Pretnar: An Introduction to Algebraic Effects and Handlers. Electronic Notes in Theoretical Computer Science 319 (2015) 19-35.

S

- [Steinberg et al. 2008] Steinberg, Dave, Budinsky, Frank, Paternostro, Marcelo, Merks, Ed: EMF: Eclipse Modeling Framework, 2nd edition. Addison-Wesley, 2008.

V

- [Völter 2013] Völter, Markus: DSL Engineering: Designing, Implementing and Using Domain-Specific Languages. 2013
- [Völter et al. 2014] Völter, Markus, Siegmund, Janet, Berge, Thorsten, Kolb, Bernd: Towards user-friendly projectional editors. In International Conference on Software Language Engineering 2014. 41-61. Springer.

W

- [Wąsowski and Berger 2023] Wąsowski, Andrzej, Berger Thorsten: Domain-Specific Languages - Effective Modeling, Automation, and Reuse. Springer 2023.
- [Wilhelm et al. 2021] Wilhelm, Reinhard, Seidl, Helmut, Hack, Sebastian: Compiler Design - Syntactic and Semantic Analysis. Springer-Verlag Berlin Heidelberg, 2013.