https://cyberlytics.eu/theses/criteria/

# Beurteilung von Abschlussarbeiten

**Prof. Dr.-Ing. Christoph P. Neumann**

(Nach einer Vorlage von Prof. Dr. Klaus Meyer-Wegener)

## Verfahren

Bei der Beurteilung von Bachelorarbeiten, Masterarbeiten und Dissertationen wird die Arbeit unter fünf Aspekten einzeln bewertet, die jedoch nicht gleichgewichtig sind. Die unterschiedlichen Gewichte werden dadurch berücksichtigt, dass für die einzelnen Aspekte verschieden hohe Punktzahlen zur Verfügung stehen.

| Aspekt | Punktzahl | | |
|---|---|---|---|
| | Bachelorarbeit | Masterarbeit | Dissertation |
| Schwierigkeitsgrad | 0 ... 6 | 0 ... 6 | 0 ... 7 |
| Originalität | 0 ... 7 | 0 ... 8 | 0 ... 10 |
| Wiss. Arbeitstechnik | 0 ... 10 | 0 ... 10 | 0 ... 7 |
| Stil | 0 ... 4 | 0 ... 4 | 0 ... 4 |
| Form | 0 ... 4 | 0 ... 3 | 0 ... 3 |
| Summe | 0 ... 31 | 0 ... 31 | 0 ... 31 |

In den Hinweisen zu den einzelnen Aspekten (Nr. 1 – 5) ist **jeweils** zunächst ein **Standard** definiert, dem eine **mittlere Punktzahl** entspricht. Dann folgen Gesichtspunkte, die eine Erhöhung (+, ++) oder eine Erniedrigung (–, – –) dieser Punktzahl rechtfertigen können. Dabei kennzeichnet ++ (– –) Gesichtspunkte, die bei Vorliegen entsprechender Umstände auch einfach oder doppelt berücksichtigt werden können. Die Gesamtpunktzahl ergibt sich durch Addition der einzelnen Punktzahlen.

Für Bachelorarbeiten und Masterarbeiten gibt es ein wichtiges Metakriterium für die Notenstufe „sehr gut“ (1,0 und 1,3): ① **Sind die Ergebnisse der Arbeit wissenschaftlich, international publizierbar?** Alternative Form dieses Kriteriums für angewandte Themen: ② **Wird der Erstprüfer oder Zweitprüfer die Arbeit anderen Professoren als Referenz bzw. als Einstieg in das Thema empfehlen?** Und noch eine dritte, abgeschwächte alternative Form: ③ **Wird die Arbeit im Kooperationsunternehmen zur Pflichtlektüre für andere Software-Entwickler oder Sachbearbeiter** (im Kontext des Themenfeldes; bspw. zur Einarbeitung neuer Mitarbeiter für das Themenfeld)**?** Signifikante Einschränkungen oder Zweifel an

## 2. Schöpferische Originalität

Bei der Beurteilung der schöpferischen Originalität ist nicht nur festzustellen, inwieweit der Bearbeiter der Anleitung und Führung durch den Betreuer bedarf. Es ist vielmehr selbstverständlich, dass der Bearbeiter Initiative entwickelt, d.h. aus eigenem Antrieb Schwierigkeiten aufgreift und mit dem Betreuer diskutiert. Im Fall einer Kooperation mit einem Unternehmen wird eine grundsätzliche Zufriedenheit des Unternehmens mit den Arbeitsergebnissen vorausgesetzt (Mittlere Punktzahl: 4 Punkte).

**Einzelhinweise zur schöpferischen Originalität:**

++ (M21) Ein bisher ungelöstes Problem wurde gelöst oder ein grundsätzlich neuer Lösungsweg für ein bereits gelöstes Problem angegeben.

++ (M22) Eine für die Aufgabenstellung untypische Programmtestmethode oder Beweistechnik wurde entwickelt.

\+ (M23) Die Aufgabe und die darin enthaltenen Herausforderungen werden durch den Bearbeiter als beispielhaft begriffen und der Bearbeiter vermag es – durch Argumentation – die Aufgabe einer höheren Fragestellung unterzuordnen.

\+ (M24) Durch den Bearbeiter werden grundsätzliche Fragen oder konkrete Vorgänge methodisch in ihren Ursachen erforscht, begründet und in einen Sinnzusammenhang gebracht (sowie durch Literaturrecherche belegt, dass die gleiche schöpferische und forschende Arbeit nicht bereits durch Dritte geleistet wurde).

\+ (M25) Während der Arbeit sich ergebende / andeutende Probleme wurden erkannt und verfolgt, auch wenn sie nicht unmittelbar zur Aufgabenstellung gehörten.

\+ (M26) Die Abschlussarbeit als Schriftstück (nicht als Quellcode, hierzu dient M33) und die darin enthaltenen Erkenntnisse können als Grundlage für weitere Abschlussarbeiten dienen (und selbstverständlich ist das in der Arbeit sorgfältig argumentiert).

– (M27) Der Bearbeiter geht sich ergebenden Schwierigkeiten aus dem Weg.

– (M28) Die Initiativen und Lösungsvorschläge des Bearbeiters sind bezgl. ihrer Durchführbarkeit nicht durchdacht.

– – (M29) Der Bearbeiter lässt es an eigener Initiative mangeln und bewegt sich ausschließlich in den durch den Betreuer vorgezeichneten Bahnen.

# Secondary School: Grade Definitions



Does NOT apply directly to universities from a purely legal standpoint! However, it serves as a reference for your expectation management.

- **By resolution of the Conference of Ministers of Education on October 3, 1968, the following definitions, which are still valid today, have been established for school grades:**

| | |
|---|---|
| very good (1) | The grade "very good" should be awarded when the performance meets the requirements to a significant degree. |
| **good (2)** | The grade "good" should be awarded when the performance **fully** meets the requirements. |
| satisfactory (3) | The grade "satisfactory" should be awarded when the performance generally meets the requirements. |
| sufficient (4) | The grade "sufficient" should be awarded when the performance has shortcomings but still meets the requirements overall. |
| poor (5) | The grade "poor" should be awarded when the performance does not meet the requirements, yet indicates that the necessary foundational knowledge is present and the shortcomings can be addressed in the foreseeable future. |
| insufficient (6) | The grade "insufficient" should be awarded when the performance does not meet the requirements and even the basic knowledge is so inadequate that the deficiencies cannot be remedied in the foreseeable future. |

The original text of the KMK resolution from 03.10.1968 is not directly available as a freely accessible digital document; it is primarily recorded in archives or as a bibliographic reference. The resolution can be found, for example, in the Federal Archives under the signature BArch B 304/3125 in connection with school affairs.

# Pillars of the grading spectrum



**Informal**
(Bachelor level)

- **1,0**
- **1,3**

① Are the results of the work scientifically and internationally publishable?
② Will the first or second examiner recommend the paper to other professors as a reference or an entry point into the topic?
③ Will the work be mandatory reading in the cooperating company for others?
(Users/caseworkers, software architects/developers, stakeholders/decision-makers, ...)

"To a special degree"

- **1,7**

**① Slightly mixed reading. Only partial wow effects.**
**② Related work is present, as well as clear and unfragmented requirement specifications**
**③ Use of formal notations from computer science/AI and existing automated tests for regression**
**④ The source code meets engineering requirements = "Under the hood, the software prototype is not a mess"!**

"Fully meets requirements"

- **2,0**
- **2,3**

① Little wow factor, but suitable for a master's degree
② For example, "How I Did It" works
③ Absence of kindergarten errors (!)

**⓪ Does your bachelor's thesis still undoubtedly demonstrate the ability to "independently address a problem from the relevant field using scientific methods within a specified period"? (National joint structural guidelines / Resolution of the Conference of Ministers of Education: https://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/2003/2003_10_10-Laendergemeinsame-Strukturvorgaben.pdf)**

**2.5: Master's admission threshold!**

- **2,7**

① Kindergarten errors present
② Does your work raise doubts about scientific methods, initiative, or engineering approach?
⊃ "Under the hood, the prototype is a mess," meaning it has significant qualitative deficiencies? A devaluation characteristic!
③ Does your work show clear signs of being superficial, uncritical, or thoughtless?

"Generally meets the requirements"

- **3,0**
- **3,3**

① Are there gaps in the fundamentals in parts?
② Is there a lack of substance in parts (possibly due to procrastination)?
[Among other things, a rule of thumb for substance evaluation: "The share of your own work – that is, what you have done – should comprise at least 50% of all pages"]

- **3,7**
- **4,0**

① Significant deficiencies in the fundamentals? (But necessary basic knowledge still demonstrated.)
② Significant deficiencies in scope, content, or form? (But minimum requirements still met.)
(Caution: Parts of the problem still solved independently, and the PDF is not a wreck, otherwise 5.0!)

"Deficiencies, but overall still meets [...]"

# Bachelor's Thesis → Master's Thesis




https://cyberlytics.eu/theses/criteria/

| Aspect | Expectations for Master's vs. Bachelor's |
|---|---|
| **Level of Difficulty** | A higher professional level (beyond standard lecture knowledge) is more likely to be expected [M14] |
| **Originality** | Stronger self-performance: Initiative, new paths, "fundamentally new solutions" [M21/M22 particularly via Master's theses] |
| **Scientific work techniques** | More mature handling of literature, comparison, differentiation, critical discussion, interpretation [M31–M33] |
| **Meta-criterion "Very good"** | Publishability or recommendability as a reference – this meta-criterion applies to both Master's and Bachelor's, but is more easily achieved in practice by Master's theses |

prompt

# Let me ~~google~~ that for you:

Act as a professor in computer science, lecturing about scientific writing. Provide the most common mistakes for each of these topics: title, abstract, introduction, background, related work, methods, results, discussion, future work, conclusion.

Provide also common mistakes for: bibliography, figures, tables, abbreviations, language, style, formatting, capitalization rules, grammar, typos.

# Simulated Evaluation / Feedback

# Simulated Feedback: Bachelor Thesis (with SW-Prototype)



Self-Review

Worth mentioning: A gpt-5 is a much stricter reviewer than gpt-5-turbo!

Bachelor

You are the author, performing a self-review of your bachelor thesis in Computer Science. Adopt the voice of a university professor / external examiner and write a professional, constructive review of the written report and the software prototype. Write the review in the same language as the thesis.

1. General impression: Clarity, relevance, and coherence of the thesis. Does the thesis convincingly communicate the purpose and relevance of the prototype?
2. Topic assessment: Is the described problem and the proposed software solution clear and well-motivated? Is the scope appropriate for a 50 pages thesis at bachelor level?
3. Structure and organization: Does the thesis follow a logical and academic structure (abstract, introduction, problem statement, related work, domain analysis or business analysis, functional requirements and acceptance criteria, architecture description, implementation, testing & deployment, evaluation, discussion, future work, conclusion, references)?
4. Functional requirements: Evaluate them for clarity, completeness, testability, and acceptance criteria. If missing or incomplete: explicitly say which critical requirements are missing and propose a minimal set of user stories and acceptance criteria appropriate for the project's stated scope.
5. Software architecture and design: Is the architecture explained clearly, e.g., with diagrams (components, interactions, deployment)? Are the architectural decisions justified and related to the problem statement?
6. Engineering aspects: Are testing strategies described adequately (unit, integration, system tests)? Is deployment considered (runtime environment, CI/CD, containerization, cloud, etc., depending on the prototype)? Are maintainability, extensibility, and usability addressed?
7. Results and evaluation: Does the report discuss the prototype's strengths and limitations? Are evaluation criteria (e.g., fitness-for-use, performance, reliability) clearly presented and supported with evidence?
8. Language and style: Is the language clear, academic, and technically precise? Are diagrams, tables, and figures labeled and referenced properly?
9. Suggestions for improvement: Provide constructive feedback on where the thesis could improve: requirements clarity, architecture clarity, use of diagrams, justification of design decisions, inclusion of testing/deployment details, critical evaluation, or academic writing style.

Write the review in the language that the exposé was written and in a tone appropriate for university-level academic feedback. You are allowed to criticize the work presented.

As appendix to the review draft, provide detailed comments: Provide in-depth, line-by-line or section-specific feedback, referencing equations, figures, or sections if available.

For each comment, use this template:
[Location] Section title — page X, para Y (or line-range) — Severity: {major|moderate|minor}
Issue: short description.
Recommendation: concrete fix (exact wording or steps).

Highlight missing or unclear aspects: unclear requirements, missing acceptance criteria, testing strategy not specified, deployment environment missing, architectural trade-offs not justified. List all typos, formatting issues, or grammar errors. Point out bibliography issues: missing references to tools, frameworks, or standards; incomplete citation style. Suggest improvements to clarity and style, e.g., improving diagram captions and better consistency in terminology.

Erstprüfer oder Zeitprüfer

# Code Quality

# Vibe Scripting for Prompting Utilities Preparation



≙ mini prompt ➔ not actual prompt engineering!
But usually still quite **effective for utility shell script generation…!**

Just an example!

Create a PowerShell 7 (pwsh) script that analyzes multiple Git repositories and extracts code ownership information.
Requirements:

- Use the current directory as the base directory.
- Find all immediate subdirectories that are Git repositories.
- For each repository, use `git ls-files` to enumerate tracked files.
- Filter files so that source code and configuration files are included, while documentation, media files, archives, generated artifacts, binaries, dependencies, and build outputs are excluded.
- Implement binary-file detection and skip binary files.
- For each included file, run `git blame --line-porcelain`.
- Attribute each source line to the author reported by Git.
- Support a configurable list of ignored authors, prefilled with `@("blueprint", "Blueprint", "cyberlytics")`.
- Generate one output file per author and repository using the naming pattern:
  `<RepositoryName>.<AuthorName>.codeextract.txt`
- Group extracted lines by source file and insert a header such as:
  `### File: relative/path/to/file`
- Preserve the original code lines exactly as they appear in the repository.
- Sanitize author names for use in filenames.
- Use UTF-8 throughout.
- Display progress information and handle errors gracefully.
- Produce a complete, well-commented PowerShell script.

**Now, you can dump all the source code at once (using Ctrl-C/Ctrl-V) right into a chatbot, for example, for a code review!**

# Simulated Feedback: Code-Quality Bachelor Thesis



Worth mentioning: A gpt-5 is a much stricter reviewer than gpt-5-turbo!

Bachelor

INPUT Provider (Learning Tutor) Yes ✓

Act as a German computer science professor reviewing a Bachelor Thesis software prototype. The attached source code represents the student's individual contribution, extracted via git blame. It is not self-contained; do not attempt to compile or run tests.

Context & Scope:
- Student workload: 120+ hours
- Expected size: 2500 effective lines of code (eLoC)
- Minimum threshold: 1250 eLoC (Fail condition if below)
- Exceeding expectation: 4000+ eLoC (if quality is high)
- Quality strictly overrides quantity. Craftsmanship is paramount; bonus points apply for high originality.

Provide your evaluation using the following structure:

1. Final Grade Suggestion: [German university scale: 1.0 to 5.0], placed at the very top.
2. Code Summary: Exactly two sentences explaining what the code does.
3. Aspect-by-Aspect Grading & Feedback: Grade (1.0 - 5.0) and bullet points for:
   - Functionality & Testing
   - Readability, Structure & Coding Standards
   - Code Organization, Modularity & Design Patterns
   - Error Handling & Edge Case Coverage
   - Efficiency, Scalability & Absence of Copy-Paste
   - Documentation (Comments on public interfaces/classes) and absence of compiler warnings
4. GenAI Detection: Note any indicators of unverified LLM-generated code.

Output the review directly, balancing rigorous academic standards with constructive feedback.

# Simulated Feedback: Code-Quality Master Thesis



Worth mentioning: A gpt-5 is a much stricter reviewer than gpt-5-turbo!

Master

INPUT Provider (Learning Tutor) Yes ✓

Act as a German computer science professor reviewing a Master Thesis software prototype. The attached source code represents the student's individual contribution, extracted via git blame. It is not self-contained; do not attempt to compile or run tests.

Context & Scope:
- Student workload: 300+ hours (Major engineering/research component of a Master Thesis)
- Expected size: 4000 to 5000 effective lines of code (eLoC)
- Minimum threshold: 2500 eLoC (Fail condition if below, unless extreme algorithmic complexity compensates)
- Exceeding expectation: 6000+ eLoC (if quality and architectural integrity are high)
- Quality and scientific depth strictly override quantity. High originality and algorithmic innovation grant bonus points.

Provide your evaluation using the following structure:

1. Final Grade Suggestion: [German university scale: 1.0 to 5.0], placed at the very top.
2. Code Summary: Exactly two sentences explaining the core architecture and purpose.
3. Aspect-by-Aspect Grading & Feedback: Grade (1.0 - 5.0) and bullet points for:
   - Scientific Value & Originality (Innovation, algorithmic complexity, or novel application)
   - Architectural Design & Modularity (Design patterns, scalability, system boundaries)
   - Functionality & Testing (substantial quality control including security considerations)
   - Readability, Code Hygiene & Standards (Clean code principles, idiomatic language use)
   - Robustness & Edge Cases (Advanced error handling, concurrency, resource safety)
   - Efficiency & Performance (Optimizations, absence of copy-paste/anti-patterns)
4. GenAI Verification: Note any indicators of unverified LLM-generated code.

Output the review directly, applying rigorous, master-level academic and software engineering standards.