Phasen, Techniken und Strategien der Softwareentwicklung an Hand eines Kursprojekts im Modul SWE-II*

*Thema Vorgabe laut Checkliste zum Kursprojekt im Modul Software-Engineering-II

Fabian Schiemann
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_schiemann21@stud.hwr-berlin.de

Oskar Hugo Thaute
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_thaute21@stud.hwr-berlin.de

Sharleen Schnelle
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_schnelle21@stud.hwr-berlin.de

Noah Reckhard
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_reckhard21@stud.hwr-berlin.de

Laurin Pausch
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_pausch21@stud.hwr-berlin.de

Minh Hai Nguyen
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_nguyenmi21@stud.hwr-berlin.de

***Abstract*—Die Theorie der Softwareentwicklung bietet zahlreiche strukturierte Informationen und Vorstellungen von Form und Struktur eines Softwareprojektes. Die Realität innerhalb eines solchen Projektes kann davon jedoch stark variieren. Sei es der Vorgesetzte in einem alteingesessenen Familienunternehmen, der Scrum-Sprints verplant, oder der rüstige Teamleiter einer großen Abteilung, der alle seine Projekte in PASCAL geschrieben haben möchte. Dieser Artikel präsentiert eine solche Variation und zeigt, wie sich eine Gruppe von sechs Studenten mit hohen Ambitionen und frisch erlangtem Wissen einem solchen Softwareprojekt widmet. Generationsbedingt wird im folgenden die Entwicklung eines Computerspiels mit dem Namen "Harsh Politics" vorgestellt. Ziel dieses Artikels ist nicht nur die Darlegung und Erläuterung spezieller Methoden und Strategien aus der Softwareentwicklung, sondern es sollen auch Antworten auf folgende Fragen an ambitionierte und ahnungslose Jung-Entwickler gefunden werden: Wie macht man ein Spiel? Kann ich das alleine machen? Wie finde ich eine gute Spielidee? Ist meine Spielidee gut? etc...**

***Index Terms*—component, formatting, style, styling, insert**

I. Einleitung / Motivation / Einführung / Hintergründe

Grady Booch gibt in seinem Paper "The History of Software Engineering" eine gute Übersicht über die Entwicklung des Begriffs "Software Engineering". Im folgenden Absatz ist das Paper zusammengefasst. Laut Booch wurde der Begriff erstmals 1963/64 von Magaret Hamilton erwähnt. Als damalige Lead-Developerin in Skylab und Apollo wollte sie ihre Arbeit von "Hardware Engineers" unterscheiden. In den 1960er soll es drei große Faktoren gegeben haben, die die Softwareentwicklung beeinflussten, und zwar die Kommerzialisierung von Software als Produkt, die Entwicklung des SAGE-Verteidigungssystems und das USRaumprogramm und die daraus resultierenden überlebenswichtigen Softwaressyteme. Zwischen den 60er und 80er wurden die Konzepte wie Modular Programming, Wasserfall Modell und Entity-Relationship-Modell für die Softwareentwicklung entwickelt. In den 80er und 90er veränderten sich die Ansprüche auf Softwaresysteme durch die zunehmende Globalisierung. Daher mussten neue Konzepte wie UML, Spiral Modell, OOP eingeführt werden, um die wachsenden Anforderungen zu bewerkstelligen. Durch das Internet in den 90er erhielt die Rolle der Software abermals eine neue Stellung. Der "Consumer" hatte nun als neuer Stakeholder andere Ansprüche an die Software als Unternehmen. Zu dieser Zeit haben sich bereits Techniken der Softwareentwicklung etabliert, wie "ContinuousIntegration" und iteratives Entwickeln. Hinzu kamen nun auch Design Pattern auf Architekturund Code-Ebene. Das Internet-of-Things gewann durch Smartphones an Bedeutung, sodass monolitische Software durch Microservices, Webtechnologien und Cloud Computing ersetzt wurden. (Booch, 2018) Wie zu sehen ist, hat das Wort "Software Engineering" eine lange

Entwicklungsgeschichte. Sie zeigt, dass durch die wachsenden Anforderungen und die wachsende Komplexität, Software gut organisiert, gemanagt und durch geeignete Richtlinien, wie Patterns, Agiles Programmieren und gute Dokumentation entwickelt werden muss. Die moderne Softwareentwicklung baut auf fünf Phasen auf. Die Analyse, das Design, die Codierung, das Testen, und die Wartung. (Sommerville, 2011) Jeder dieser fünf Phasen nutzt Techniken und Strategien, die in den letzten 60 Jahren entwickelt wurden. In der Analyse gibt es Stakeholderlisten und Anforderungsanalyse, aus der sich funktionale und nicht funktionale Anforderungen ableiten lassen. Aus diesen lassen sich das Design entwickeln, in dem man aus den Anforderungen UML-Diagramme und Softwarearchitekturen entwickelt. Hinzu kommen Grundsätze und Leitsätze, die die Gestaltung von Bedienoberflächen und Dialogen festlegen. Dadurch lassen sich den Stakeholderlisten nahe Software entwickeln. In der Implementierungsphase haben sich bestimmte Frameworks, und Code-Richtlinien, wie YAGNI, DRY, KISS, SOLID durchgesetzt, um eine hohe Qualität des Codes zu gewährleisten. (DRY vs KISS – Clean Code Prinzipien, 2018; t2informatik GmbH, 2022; Oloruntoba, 2021) Genauso ist das Testen und Warten von Software sehr wichtig. Software spielt in vielen Lebenslagen der Menschen eine wichtige Rolle und würde bei einem Ausfall zu fatalen Folgen führen. Um die Software zu verbessern, werden daher auch oft Refactoring betrieben. Für die Realisierung von Software werden dann auch sowas wie Agiles Programmieren, Scrum und Kanban verwendet. Das Ziel dieses Projektes ist es zu zeigen, dass durch eine gute Planung und unter Einhaltung moderner Programmier-Richtlinien eine releasefähige Software im Rahmen des Studienprojektes in einem angemessenen Zeitraum erstellt werden kann. Dabei wurde für die Entwicklung eines Spiels namens "Harsh-Politics" entschieden. Das Spiel baut auf altbekannte Spiele auf, wie Tekken und andere diverse Kampfspiele, in der zwei Spieler gegeneinander kämpfen und unter anderem verschiedene Fähigkeiten besitzen. In diesem Projekt sollen bekannte Politiker gegeneinander kämpfen. Dabei besitzt jeder Politiker typische Wiedererkennungsmerkmale, die unter anderem auch als Spezialfähigkeit eingesetzt werden kann. Mit diesem Spiel werden die Phasen der Softwareentwicklung in den folgenden Kapiteln beispielhaft verdeutlicht.

II. Anforderungen

Als relevante Stakeholder haben wir zum einen Maria, als eine Art Kundin oder Auftraggeberin identifiziert. Zum anderen, die späteren Nutzer, also die Spieler und außerdem unser Team, als Entwickler. Als wichtigste Projektziele wurden folgende Anforderungen ermittelt:

- Das wichtigste Projektziel ist die Lauffähigkeit des Programms. Hauptquelle dieser Anforderung ist Maria als Stakeholderin. Das Ziel ist insofern von großer Relevanz, als dass es ihre einzige Anforderung ist. Sie nicht umzusetzen wäre also fatal.
- Als weiteres allgemeines Projektziel haben wir Entwickler uns nahezu von Beginn an auf die Anforderung geeinigt, dass es bei Programm um ein zweidimensionales Spiel, im eins gegen eins Modus handeln soll.

Sowohl funktionale als auch nicht-funktionale Anforderungen wurden gemeinsam als Team ermittelt, indem jedes Mitglied mithilfe von Brainstorming seine speziellen Wünsche ins Spiel einbrachte. Außerdem wurde sich in die späteren Nutzer hineinversetzt und aus ihrer Perspektive wurde überlegt, was für das Programm wichtig sein könnte. Die Anforderungen entstanden nicht alle zu beginn, stattdessen kamen beispielsweise auch noch während des Entwicklungsprozesses weitere dazu. Darüber hinaus brachte auch das ermitteln möglicher Risiken, einige neue Anforderungen ein.

A. Nicht-Funktionale Anforderungen

- Auswahl verschiedener Charaktere: Die Spieler sollen die Möglichkeit haben, das Aussehen und Können, also die Fähigkeiten, ihrer Figuren zu ändern, also aus verschiedenen Charakteren zu wählen.
- Mapauswahl: Neben der Auswahl verschiedener Charaktere soll Spieler auch Möglichkeit haben zwischen verschiedenen Maps zu wählen, also verschiedenen Spielfeldern.
- dynamische Animationen und Darstellungen: Verschiedene Darstellungen und Animationen sollen das Spiel dynamischer und interessanter machen. Beispiele dafür Sprung- oder Angriffsanimationen oder auch die Spiegelung der Charaktere je nach Richtung in die sie laufen.
- Skalierbarkeit: Als weitere Anforderung wurde die Skalierbarkeit der Anwendung gewünscht. Sie soll in eine gewünschte Größe gezogen oder an die Größe verschiedener Bildschirme angepasst werden können.
- Zuverlässigkeit: Die Anwendung muss zuverlässig funktionieren, soll möglichst flüssig laufen und nicht abstürzen.
- Benutzerfreundlichkeit: Der Nutzer sollte sich schnell und einfach im Programm zurechtfinden können. Dafür muss die Steuerung möglichst selbsterklärend und leicht verständlich sein, um ein optimales Spielerlebnis zu gewährleisten.

B. Funktionale Anforderungen

- Steuerbarkeit der Figuren: Damit die Figur sich nach Anweisung des Spielers bewegen kann, muss eine Eingabe auf der Tastatur möglich sein. Die Eingabe muss vom Programm ausgewertet und umgesetzt werden können.
- zwei Spieler, mit getrennter Steuerung: Um das Projektziel des 1v1 Spiels umzusetzen, muss es zwei getrennte Figuren geben. Damit zwei Spieler gegeneinander spielen können, müssen die Steuerungen der Figuren verschieden sein.
- Figuren können springen: Der Spieler soll die Möglichkeit haben, seine Figur springen zulassen.
- Angriffe auf den Gegenspieler: Da zwei Spieler gegeneinander spielen sollen, müssen Angriffe der eigenen Figur gegen die Figur des Genengers möglich sein.

- Spezialangriffe/-fähigkeiten: Neben den normalen Angriffen sollen Spezialangriffe oder Fähigkeiten möglich sein, die je nach ausgewähltem Charakter unterschiedlich sind.
- gegnerische Angriffe abwehren: Um den Angriffen des Genengers nicht hilflos ausgeliefert zu sein, sollen die Spieler sich gegen die Angriffe verteidigen können.
- Menü für Einstellungen: Als weitere Anforderung wünschten sich die Entwickler ein Menü, indem über Buttons verschiedene Einstellungen vorgenommen werden können. Darunter zählen neben der bereits erwähnten Map- und Charakterauswahl, beispielsweise auch das Ein- oder Ausschalten der Sounds.

Die hier genannten Anforderungen sind ein Auszug aus den im Projekt ermittelten Anforderung. Sie wurden von der Gruppe als am wichtigsten identifiziert.

III. Organisation

Zu Beginn haben wir unsere Projektgruppe in die folgenden Arbeitsgruppen unterteilt: Management, Entwicklung und Design/Art. Hierbei gab es eine Person im Management, die zeitgleich im Sounddesign tätig war, drei Entwickler und zwei Artists. Der Art-Bereich war außerdem unterteilt in Environment Design und Character Design.

Ein Großteil des Projekts wurde mit agiler Softwareentwicklung erstellt. Wichtige Bestandteile hierbei waren die ”Inkrementelle Entwicklung“ und der ”Scrum“ Prozess. Der Fokus für die Entwickler lag anfangs vollständig darauf, sich mit Unity vertraut zu machen und die Basisfaktoren umzusetzen. Diese fundamentalen Funktionen und das Wissen, diese zu benutzen, waren grundlegende Kenntnisse, die benötigt wurden, um jeweils eigene Teilsysteme bearbeiten und bewältigen zu können. Hierbei gab es keine Vorgaben außer dem generellen ”politischen“ Thema des Spiels. Während der Entwicklung wurden wöchentliche Meetings durchgeführt, die den aktuellen Sprint beendeten, auswerteten und mit dem ganzen Team zusammen den neuen Sprint vorbereiteten. Die genaue Umsetzung der einzelnen Schritte war hierbei vollkommen den Bearbeitern belassen. Genauso selbst überlassen waren die Arbeitszeiten, da wir asynchron gearbeitet haben. Es war lediglich wichtig, das vorher festgelegte Pensum an Aufgaben im momentanen Sprint zu erledigen. Die Hauptkommunikation zwischen Vorlesungen und Meetings lief über Whatsapp, die Meetings selbst über Discord und der Datenaustausch über Github. Im Github-Repository haben wir eine Aktivitätenliste mit den noch anstehenden Aufgaben erstellt. Hier wurde das Projekt in drei Phasen unterteilt. Die jeweiligen Phasen wurden des Weiteren in die einzelnen Teilsysteme unterteilt, die bearbeitet werden müssen. Mit Hilfe dieses Backlogs wurde in den Meetings immer besprochen, an welchem Punkt wir uns momentan befinden und welche Teilsysteme als nächstes behandelt werden müssen. Die ersten dieser Teilsysteme, waren zum Beispiel: ”Sprungphysics“, ”zwei Dummies, die sich unabhängig voneinander bewegen können“, ”getrennte Steuerung“ und ”ein langer schmaler Background“. Zudem wurde innerhalb der Meetings festgelegt, welche Aufgaben des letzten Sprints nicht zur vollständigen Zufriedenheit erledigt wurden. Diese wurden dann in den nächsten Sprint mit aufgenommen. Danach wurde bestimmt, welche Teilsysteme für welches Departement die höchste Priorität haben. Anschließend wurde der nächste Sprint eingeleitet. Bevor Aufgaben als erledigt gelten konnten, war es wichtig, dass sie mit der Gruppe geteilt werden. Daraufhin hat die Gruppe ihr Feedback dazu gegeben und dieses musste umgesetzt werden. Erst dann sind sie vollständig abgeschlossen und der Bearbeiter kann sich einem neuen Teilsystem widmen.

IV. Umsetzung

A. Kampsystem

Das Kampfsystem funktioniert aufgrund eines Lebenssystems. Jeder Spieler hat eine bestimmte Anzahl an Leben, die aufgebraucht werden müssen, damit er besiegt ist. Um diese Leben auf 0 zu reduzieren, stehen jedem Spieler verschiedene Waffen zur Verfügung. Dabei handelt es sich zum Beispiel um Schwert und Schild, aber auch um eine Pistole die wirksam auf größere Reichweite ist. Zusätzlich zu diesen Waffen hat jeder Spieler abhängig von seinem gewählten Charakter zugriff auf verschiedene Spezialfähigkeiten, die er nach Ablauf eines Cooldowns aktivieren kann.

B. Movementsystem

Um die Spieler in der Szene zu bewegen greifen wir per Skript auf eine vordefinierte Funktion von Unity zu, die mithilfe einer geschwindigkeitsvariable und vordefinierten Inputs bezüglich der Richtung die Spielfigur mir der festgelegten Geschwindigkeit in die gewünschte Richtung bewegt. Das Springen funktioniert ähnlich. Statt einer horizontalen Koordinaten Änderung gibt es hier eine vertikale. Um die Schwerkraft kümmert sich Unity wieder selbst, da jedes Objekt welches eine Komponente namens Rigidbody2D besitzt auch der Physik Engine unterliegt. Wie hoch die Spielfigur bei einem Input springt wird wieder durch eine zusätzliche variable entschieden, die man hier jedoch als Sprungkraft bezeichnet. Um sicherzustellen, dass kein spiel Objekt durch den festgelegten Boden fällt wird eine Funktion angewandt die mithilfe eines körperlosen Punkts unter dem Objekt feststellt ob diese sich auf einem festgelegten Boden befindet oder nicht.

C. Animationen

In dem Spiel sind verschiedene Objekte animiert. Dabei handelt es sich um die Spielcharaktere selbst, sowie um ausgerüstete Waffen. Wichtig dabei ist, dass Waffen die z.B. auf dem Boden liegen keine Animation haben. jedes dieser animierten Objekte besitzt eine Idle Animation, also eine Animation solange sich das Objekt im Ruhezustand befindet. Sei es bei den Figuren wenn sie bewegungslos stehen, oder bei den Waffen wenn sie ohne anzugreifen in der Hand gehalten werden. Die Spielfiguren besitzen des Weiteren lauf-animationen und Sprung Animationen. Die Angriffsanimationen sind je nach ausgerüsteter Waffe unterschiedlich und werden deshalb in den Waffen ausgelöst. Übergänge zwischen diesen verschiedenen Arten von Animationen (Idle, Bewegung/ Idle,

angriff) werden mit einer Trigger variable ausgelöst. Sobald eine bestimmte Trigger variable ausgelöst wird, z.B. durch Tastatur Eingabe, löst ein an das Objekt angehängter Animationscontroller die jeweils geforderte Animation aus.

D. Aufbau des Spielfelds

Nach Start des Spiels wählt jeder Spieler sich einen Charakter aus, den er spielen möchte. Um diese verschiedenen Charaktere in verschiedenen Szenen spielen zu lassen gibt uns Unity die Möglichkeit vorgefertigte Spielobjekte zu erstellen, welche man immer wieder in die vorhandene Szene kopieren kann. So ist es möglich Charaktere und Spielfelder mit vorgefertigten Funktionen, Fähigkeiten und Hindernissen immer wieder neu und in jeglicher Kombination aufzubauen.

E. Waffen

Das System, um Waffen aufzuheben funktioniert in dem Spiel auf dem Prinzip einer Hand des Charakters. Die Hand ist in diesem fall ein körperloser Punkt, der auf eine bestimmte Position abhängig von dem Charakter gesetzt wird und als ankerpunkt für jegliche Items dient. Zu beginn des Spiels ist dieser hand-ankerpunkt von einem Objekt namens Hand belegt, welches als Platzhalter dient, aber auch für den Kampf benutzt werden kann. Wenn der Spieler mit dem Hand-ankerpunkt über einer Waffe steht und den Befehl fürs aufheben gibt, wird dieser Hand Platzhalter durch die jeweilige Waffe mit ihren Eigenschaften und Animationen ersetzt. Bei der Funktion die Waffe wieder wegzuwerfen, wird die Waffe dann wieder durch den Hand Platzhalter ersetzt. So sollte es dem Spieler unter normalen umständen unmöglich sein unbewaffnet oder mit leerer ”Hand“ dazustehen.

V. Fazit

In diesem Paper wurde die Methodik und das Vorgehen für das beschriebene Projekt ausführlich dargelegt und veranschaulicht. Die Zusammenarbeit im Team und die eigene Konzeptionierung, halfen den Studierenden dabei, die Erfahrungen aus dem vorherigen Semester im Bereich Softwareengineering zu vertiefen und bot zugleich einen Ausblick darauf, wie Softwareprojekte im späteren Berufsleben ablaufen könnten.

Während der Projektarbeit mussten die Studierenden lernen, ihre eigenen Ansichten und Gedanken, anderen Teammitgliedern gegenüber zu verteidigen und Kompromisse zu finden, um das Vorhaben voranzubringen. Zudem mussten sie, aufbauend auf dem Modul ”Softwareengineering 1“, verschiedene Phasen der Softwareentwicklung durchlaufen und selbstständig Lösungen für Probleme finden. Sie erhielten vertiefte Einblicke in die Modellierung von Klassen- und Sequenzdiagrammen in der Unified-Modeling-Language (UML) und lernten, wie man ein Produkt mittels Projekt-Teasern anschaulich gegenüber eventuellen Kunden vorstellt.

Jedem Gruppenmitglied wurden feste Aufgabengebiete zugeteilt, wie beispielsweise die grafische oder die technische Umsetzung, unter Berücksichtigung der jeweiligen Stärken und Interessen. Dadurch sollte die Arbeit am Projekt einerseits asynchron und andererseits möglichst effizient stattfinden. Die grundsätzliche Projektplanung wurde gemeinsam als Gruppe durchgeführt und aufgetretene Probleme wurden schnell angesprochen und als Team gelöst. Meinungsverschiedenheiten wurden ebenfalls in der Gruppe diskutiert, um diese möglichst schnell zu beseitigen.

Zusammenfassend lässt sich sagen, dass das Projekt einen großen Mehrwert für die Studierenden und eine Vorbereitung auf den zukünftigen Arbeitsalltag mit sich brachte.

Die Unterteilung des Projektes in einzelne Phasen stellte sich im Nachhinein als sehr hilfreich heraus, da somit das gesetzte Ziel eines lauffähigen Prototyps relativ früh im Entwicklungsprozess erreichbar war. Zwar enthielt dieser Prototyp zunächst nur grundlegende Funktionen, wie die Steuerung oder Sprünge, jedoch war dies ein erster Meilenstein, der die nächste Projektphase einleitete. Schritt für Schritt wurden so weitere Funktionen implementiert, die schlussendlich zum fertigen Produkt weiterentwickelt wurden.

Die Entwicklungsumgebung der Unity-Engine, die essenziell für die Entwicklung des Spiels war, generierte zu Beginn des Projekts einige Probleme, da es ein gänzlich neues Tool war, in das sich erst eingearbeitet werden musste. Mit der Zeit wuchs jedoch das Verständnis für die Materie und es wurden erste Testkonfigurationen an vorgefertigten Beispiel-Projekten durchgeführt. Schließlich wurde das angeeignete Wissen gesammelt und Verständnislücken innerhalb der Gruppe geschlossen.

Umfassende Gespräche halfen dabei, Dinge besser zu verstehen und trugen zur Projektplanung bei. Wöchentliche Meetings brachten neue Meilensteine hervor, die gesetzt und schließlich durch neue Ziele ersetzt wurden.

VI. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections VI-A–VI-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as “3.5-inch disk drive”.

- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for

these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.