

Humorous Songs to Accompany the Teaching of Software Engineering

Henry Gardner

Research School of Computer Science

The Australian National University

Canberra, ACT 2600, Australia

Henry.Gardner@anu.edu.au

Abstract—A collection of humorous songs has been written to accompany the teaching of software engineering. These songs have been written to reflect the typical phases of a university software engineering course and have been structured in increasing levels of difficulty so that beginners can learn to play them on the ukulele. The songs have improved student enjoyment of a large software engineering course that has been delivered during the anxious times of the covid19 pandemic.

Index Terms—humor, education, software engineering, song, singing, cartoon

I. INTRODUCTION

This experience paper describes an approach to the teaching of software engineering that involves a parallel activity where students are taught to sing (and play on the ukulele) a set of humorous songs that have been especially written to accompany the course material. After centuries of being deprecated, the use of humor in education is now regarded as one of the traits of a good teacher [1]. Academic studies of humor in education abound, including some that demonstrate that a humorous approach can not only improve student appreciation but can result in improved academic outcomes [2].

Arguably, there has never a greater need to lift the spirits of university students (and their teachers). At the time of writing, the world is experiencing a period of great disruption and anxiety because of the covid19 pandemic. Students are understandably anxious about their health and well-being and the health of their friends and families. They are understandably anxious about their future careers in a pandemic-disrupted world. It could be that the use of gentle, *non-hostile* humor has a comforting role that can complement the subject matter for these students. In a wider sense, some medical practitioners recommend that some humor and laughter can be helpful for “these trying times” [3].

In Section II, we briefly review humor and education, and then we reflect on what it is about software engineering that makes it funny (Section III). Our songful approach is described in Section IV and the Appendixes. The student reception to our songs is discussed in Section V together with a reflection on its success relative to an earlier, cartooning approach (in Section VI). The paper concludes with Section VII.

II. HUMOR AND EDUCATION

Serious consideration of humor in philosophy and education can be traced back to ancient times. Plato is said to have

expressed very negative views about laughter and comedy. As discussed by Morreall [4], Plato strongly argued against the teaching and performance of comedy because laughter involved holding a person, object or idea up to *hostile* ridicule: “No composer of comedy, iambic or lyric verse shall be permitted to hold any citizen up to laughter, by word or gesture, with passion or otherwise” (Plato’s Laws 11:935 [5] as cited by [4]). Such views strongly influenced the early Christian church and, through it, education and philosophy for centuries. In fact, it is only since the 1800’s that a sense of humor has been appreciated as an attractive personality trait [4]. This revision in thinking can be traced to the erosion of the power of the church and the recognition by enlightenment philosophers that laughter and humour can hold up a *non-hostile*, incongruous mirror to established thought patterns to enable us to appreciate the world in different ways.

The use of humor in education has found a place in modern pedagogy. In his seminal work, “The Art of Teaching”, Gilbert Highet lists humor as one of the desirable qualities of a good teacher [1]. Since the 1970’s there have been a number of empirical studies that have looked at the impact of humor on students’ perception of teaching and a meta-analysis of the findings of such studies has been provided by Martin et al [6]. A wider treatment of the concept of humor across all areas of psychology and its applications to areas including education is the topic of a widely-cited book by Martin and Ford [7].

Among the empirical studies of the use of humor in university education have been field studies [8], surveys [9] and controlled experiments [2]. The experimental study by Ziv [2] found that students who experienced a “humor” condition in the teaching of statistics and psychology obtained significantly higher scores on the final exams in those subjects than non-humor control groups.

Some publications provide specific examples of humor in academic instruction. Lomax and Moosavi [10] provide humorous examples that have been obtained from their own practice in the teaching of statistics, from a survey of statistics teachers, and from searching the internet. These examples include jokes, humorous proverbs, “bumper stickers”, “movie titles” and “rap poems”. They record that, in their own experience, the use of humor can lessen student anxiety, motivate students to learn and can help move students to a deeper level of conceptual understanding.

Rap-poetry and word play allow a bridge from joke-oriented humor in education to the use of song and music (as a modern echo to Plato's dread of comedy and iambic or lyric verse). Amongst recent studies of the use of humorous song in educational pedagogy, Tisdell [11] has described the development and evaluation of a song called "e is a Magic Number" to enhance the teaching of mathematics.

III. WHY IS SOFTWARE ENGINEERING FUNNY?

Software engineering is a technical and management discipline concerned with working in teams to solve complex problems through the development and deployment of software and technology. The workplace settings in which software engineering is embedded affords plenty of scenarios for humor. A typical internet search on "software engineering jokes" returns thousands of results.

Perhaps the most famous examples of humor related to software engineering are the Dilbert cartoons. Dilbert is widely considered to be funny and he and his colleagues appear either to be software engineers or to work in closely allied engineering professions. Dilbert cartoons are frequently used by instructors and organisations to illustrate principles of software engineering (see for example [12]). Scott Adams, the author of the Dilbert cartoon series, describes in his book "The Dilbert Principle" [13] that the absurdity of business is due to the fact that we are all "idiots" about different things and at different times of the day. When one adds to this absurdity the customs and mannerisms of "engineers", that Adams claims to have observed first-hand much as a "primatologist studies the great apes", and you have a funny comic strip.

Closer to the substance of our software engineering course, the author encourages students to access and reflect on videos presented at developer conferences. Such videos can be quite funny, particularly those presented by renowned eccentrics such as Robert C. Martin ("Uncle Bob"). In Bertrand Meyer's important critique of agile software engineering [14], he employs hostile humor to satirize some of the excesses of some agile evangelists [14]. In particular, Meyer's deconstruction of the famous photograph of the signatories of the agile manifesto ("a half-dozen middle-aged, jeans-clad, potbellied gentlemen turning their generous backs to us"¹) is an example of hostile humor that would have upset Plato a great deal.

In software engineering education, a Dilbert-themed game developed at Lockheed Martin Corporation has been used to teach ethics in engineering (including software engineering) [15]. Outside of Dilbert, the use of humour in software engineering education has received some attention in the literature. In a recent paper in these conference proceedings, Baniassad showed that a humorous graphic was helpful in teaching the Liskov Substitution Principle [16].

Both in its technical and management aspects, software engineering enjoys lashings of jargon which render it perfect for rap-poetry and lyrical verse. This is the principal approach to humor that is employed in the songs written for our course.

Although there are some videos of humorous songs related to software engineering and programming on the internet, they appear to be rather small in number, relatively recent, and not particularly oriented towards education.

IV. A SONGFUL APPROACH

This experience paper relates to a software engineering course that has been taught in two simultaneous versions, to both second-year undergraduate students and masters by coursework students, since 2018. It is a revision of a previous course that has been taught by the author since 2016. The undergraduate version of the course assumes that students have completed at least two programming courses with a pre- or co-requisite of a third programming course. Two of the three programming courses will have been in an object-oriented language. This software engineering course is purely theoretical and modelling based. It does include a major group assignment but this is one of modelling and design rather than implementation. In the structure of the undergraduate degree, the course acts as a bridge between the programming courses and a major, year-long, group software engineering project that students undertake in their third year. Combined class enrolments over the three years of this course have been very large at 257, 421 and 422 for 2018, 2019 and 2020 respectively. Well over half of the students in the course are international students and, with the advent of the covid19 pandemic, the course is presently being presented fully online in 2020 with many students located overseas.

This software engineering course adopts a familiar format of starting with an introduction to software engineering process and then proceeding with segments that mirror the major phases of requirements, design, testing, maintenance and evolution. Software construction is not emphasised apart from references to it from the other segments. The Masters version of the course specifically considers the methodology of Domain Driven Design [17]. The course is compulsory for many of the undergraduate students and, for many, it will be the final software engineering course that they take apart from a third-year course in software project management and the year-long group project. Thus, the pressure is on to provide sufficient material for many students to take away and use in their projects and to reflect on in their future careers. Many students find the material in the course to be challenging and different to their other programming-based and mathematically-based courses. The heavy use of language and the widespread ambiguities involved in the discipline are frustrating for many international students whose first language is not English. Many students are anxious about their grades and work very hard. As remarked in the introduction, in this time of the covid19 pandemic, many students are anxious about their health and the health of their loved ones, as well as their future careers.

The songs written to accompany the teaching of this course mirror the path of students through the educational material. Seeing as they have transitioned to this software engineering course from introductory programming we start with a song

¹Page 1 of [14]

about programming, originally called “Joy of Compilation” and now called “Java Compilation” (Appendix A). This is a simple patter song based on a folk tune and contains familiar, but salutary advice such as

If there’s no semi-colon at the end of the line,
You get a compile error every time

Following the segment on process, students are introduced to a first process song, “SWE Song” which describes an engineer’s supposed experience with an agile project (Appendix B). This song goes to the tune of “Swing Low Sweet Chariot” by Wallace Willis and its wordplay affords a whimsical reference to scrum:

SWE Song, sweet agile process,
Scrumming for to carry me home.

Students then experience a three week segment on requirements engineering. Our requirements song is based on the Gilbert and Sullivan song “A Paradox” from “The Pirates of Penzance” (Appendix C). Once again, salutary and world-weary advice is offered such as:

Somewhere in the repository
Lay the answers to our woes
If tests had traced them from inception
All would have been fine.

As students process to the design part of the course, they are treated to a song on Domain Models that goes to the tune “Hey there, big spender” by Cy Coleman and Dorothy Fields (Appendix D):

The minute we drew up the box
It was clear this was a class of distinction
A real core concept.

Students then study design patterns with an accompanying song that goes to the tune of “Moon River” by Henry Mancini (Appendix E):

Design Patterns
Provide a sense of style
Of Object Oriented style
For code.

There are other songs as well: a warm-up song to the tune of Pachelbel’s Cannon that is sung in binary (!), a song about a project that is out of control and some folk songs. A “capstone song” is a self-composed anthem to a fictitious software engineering process that combines the Unified Process (UP) with an agile approach that includes a major pause in the construction phase to implement major “usability refactorings” (URS), and also some “sneaky design”. This new process is called “UP-URSSd”. You can pronounce it as you wish...

V. EXPERIENCES AND STUDENT RECEPTION

Humorous songs have accompanied the delivery of the software engineering course in 2019 and 2020. In 2019, an initial set of songs was written and orchestrated for in three-part harmonies. A choir, called the “SWE Singers” was formed before the course began and students in the course were invited to join in. The choir gave several public performances and

videos were made several of which can be viewed on the author’s home page [18].

In 2020, it has been impossible to convene the SWE Singers due to mandated social distancing. In this year, the entire course has also been put fully online. In order to keep the music going, the author has rewritten the songs, and composed some new ones, to be played on the ukulele. Live-streamed performances of songs have then been a regular part of the course in a weekly tutorial session. The author is a beginner on the ukulele and the songs themselves have been written so that they increase in complexity as the course proceeds from three chords (in Java Compilation) to eight chords (in Design Patterns). In addition to the live streaming tutorials, some of which can be seen on the author’s home page [18], a small group of students has met on a regular basis with the author to play the songs together.

There is no doubt that the inclusion of this humorous musical thread has enhanced the morale of students and contributed to a good esprit d’corps in the course. The reception to streaming ukulele performances has been very enthusiastic with lots of likes and positive icons and comments such as “Amazing”, “It’s sooo good”, “That was awesome” and “Literally the best part of my day”. Several students have purchased ukuleles and are practising away, possibly much to the annoyance of their families. A straw survey conducted of the students for this publication received 23 responses with 20 saying that the songs had increased their enjoyment of the course (3 had no opinion). Of these 23 survey recipients, 11 said that they sung the songs themselves and 10 said that they had ukuleles.

It is the opinion of the author that the students liked the ukulele songs more than the choir approach. The choir is spectacular, and relatively high-culture, but the students seemed to like seeing their lecturer making an effort to play and sing the songs himself. Perhaps in future years, both modes of delivery will be trialed together.

In spite of the positive reception from the student community, the actual numbers of students who have volunteered to join the choir or show up for the weekly ukulele session are small; just a handful out of a class of over 400. It is perhaps too much to expect anything more given the pressure that students are under, particularly in the present time. However, the good feeling seems to permeate the entire class. A deeper question is whether the exposure to the language in the songs actually helps the students in their study of software engineering. Intriguingly, 14 of the 23 respondents to the straw poll said that the songs *had helped with their study* of the course.

VI. SONGS ARE BETTER THAN CARTOONS!

In the experience of the author, there have not been any seriously negative comments from students about the humorous songs. This is in contrast with an earlier approach to incorporate some humor in the teaching of a predecessor course (“Software Analysis and Design”) in 2016 and 2017. In that earlier course, the author drew his own cartoons to illustrate the lecture slides. These cartoons were not inspired

by Dilbert but by the very popular *Discovering Statistics using [SPSS, R, SAS]* books by Andy Field [19], [20]. Many people, including this author, find these texts to be wildly funny and a breath of fresh air in the teaching of statistics. Field illustrates his books with his own cartoons built around personas such as “Smart Alex”, “Oliver Twisted” and “Labcoat Leni”. These personas take on particular attitudes that might characterise sub-groups of the student population. Their questions and comments can be quite funny and provocative and they can stimulate readers to think about the material in different ways.

In the case of the author’s Software Analysis and Design course, it was felt that drawing cartoons might not only amuse and stimulate students but that it might also inspire them to hand-draw their own UML-like diagrams. Such an agile approach to hand-drawing UML is advocated in the book by Larman [21] that was used as a text for this earlier course. The author’s cartoon personas comprised an enthusiastic lecturer (like the author), a sarcastic student, a dead body (drawn as a chalk outline with a nod to the Software Engineering *Body of Knowledge* [22]) and some chickens (with a nod to Martin *Fowler*). As well as their static representations on lecture slides, some short “puppetry” videos were made of these cartoon characters. Some students loved them and provided feedback such as “After reading SWEBOK I felt just like that chalk outline!”. However, feedback from others were that they found the cartoons to be distracting from their revision of the lecture slides at exam time. One student expressed the view that the cartoons were making fun of them. It is perhaps the case that this earlier cartooning humour was too “hostile” for this student population. Following this negative feedback, the cartooning approach has been completely discarded.

VII. CONCLUSION

This paper has described a course that complements the teaching of software engineering with a collection of humorous songs. The songs have been written to accompany different parts of the course and they have been structured to enable beginners to learn them on the ukulele. The songs have helped to lift spirits and improve the enjoyment of learning during a difficult year that has been dominated by the covid19 pandemic. There is some intriguing evidence that the songs have even helped some students with their study of the course.

The songs described here and on the author’s home page [18] are only a beginning. Some students have suggested their own lyrics. The author welcomes their contributions and those of all educators, enthusiasts and software engineering academics. Over time, perhaps we could build up a Software Engineering Body of Teaching Songs (SWEBOTS) to accompany and promote the teaching of our discipline.

APPENDIX A LYRICS FOR “JAVA COMPILATION”

This song is sung to a traditional Australian folk tune called “The Black Cat Piddled in the White Cat’s Eye”.

You have to put a semi-colon at the end of a line.

You have to put a semi-colon at the end of the line.
If there’s no semi-colon at the end of the line,
You get a compile error every time.

So many syntax errors that we wish we hadn’t done.
So many syntax errors, it’s frustrating everyone.
The code that we checked out from that Github link
online,
Now gives a compile error every time..

I’ve lost a curly bracket and my block is out of scope.

I’ve lost a curly bracket and my block is out of scope.

So I type a curly bracket and for the best I hope,
But my efforts won’t compile and my plans go up
in smoke!

So many syntax errors ...etc

My sub-class extends super and calls a method there.
My sub-class extends super and calls something
somewhere.

But there is no resolution for that method invocation,
And the compiler has a mouthful that denies
interpretation

So many syntax errors ...etc

You have to put a semi-colon at the end of a line.
You have to put a semi-colon at the end of the line.
If there’s no semi-colon at the end of the line,
You get a compile error every time.

APPENDIX B LYRICS FOR “SWE SONG”

This song is sung to “Swing Low Sweet Chariot” by Wallace Willis.

SWE song sweet agile process,
Scrumming for to carry me home.
SWE song sweet agile process, Scrumming for to
carry me home.

I went to the backlog and what did I see?
Scrumming for to carry me home...
A whole lot of user stories waiting there for me.
Scrumming for to carry me home...

SWE song sweet agile process... etc

I went to the scrum master; what did she say?
Scrumming...etc
Wait until tomorrow, and that bug will go away!
Scrumming...etc

SWE song sweet agile process... etc

We did regression testing; what did we find?
Scrumming...etc
Continuous integration left all my code behind!
Scrumming...etc

SWE song sweet agile process... etc.

APPENDIX C LYRICS FOR “REQUIREMENTS PATTERN(S)”

This song is sung to “A Paradox” by Gilbert and Sullivan
(from the operetta “Pirates of Penzance”).

No tests our software could survive.
The customer would not sign off. The team was
angry and frustrated.
It was time to stop!

Somewhere in the repository
Lay the answers to our woes.
If tests had traced them from inception
All would have been fine

Requirements, requirements, all software needs re-
quirements
Ha ha ha ha ha ha ha ha
Re-qui-re-ments
Requirements, requirements, all software needs re-
quirements
Ha ha ha ha ha ha ha ha
Re-qui-re-ments

Though agile process be in fashion
Daily stand-up meetings call
Your coding efforts might be wasted
When put to the test
Requirements change throughout a process
Even so they must be noted.
Non-functionality considered,
Metricated too

Requirements, requirements etc..

Requirements workshops are terrific
fun and entertainment too but
often need some augmentation
for business value.
So watch your contract and your client
Watch non-functional re-
quirements and make sure your testing
traces and it makes sense too.

Requirements, requirements etc..

APPENDIX D LYRICS FOR “DOMAIN MODELS”

This song is sung to “Hey there, big spender” by Cy
Coleman and Dorothy Fields (with a additional refrain).

DDD Domain
DDD Domain
DDD Domain

The minute we drew up the box
It was clear this was
A class of distinction,
A real core concept.

Domain Models,
So refined,
Go drive your business value
to the end of the line.

We’re drawing a link from that box
to another for a key relationship.
Domain Models! (Domain Models!)
Drive ...
Some business value with me.

DDD Domain
DDD Domain
DDD Domain

At one of the ends of a link
Between our boxes we can
Put little numbers
And special symbols.

Mul-ti-plicities
can be shown
The collections can be attributes
When they are known.

So let me get right to the point:
Domain Concepts should have
Names clients understand.
Domain Models! (Domain Models!)
Drive ...
Some business value with me.

DDD Domain
DDD Domain
DDD Domain

And so to the untrained eye
The model might seem to have
A lot of mess and boxes
And annotations.

Keep user interfaces away!

Put them in another layer for another day.
 So if we now start from our Domain Model
 We can design our software
 To maintain and extend.
 Domain Models! (Domain Models!)
 Drive ...
 Some business value with me.

DDD Domain
 DDD Domain
 DDD Domain

APPENDIX E LYRICS FOR “DESIGN PATTERNS”

This song is sung to “Moon River” by Henry Mancini

Design Patterns
 Provide a sense of style
 Of Object Oriented style
 For code.

Design patterns
 When carefully chosen
 Provide software structure
 That extends and maintains.

Coupling and cohesion
 Should be balanced in design
 And patterns show the way
 For code

We’re building clean architecture
 Protecting business rules
 We’re using all the tools,
 Design Patterns
 In code.

Facade and Adapter,
 Singleton and State,
 The Proxy and and factories too.

Observer and Memento,
 Strategy and Command,
 The Decorator, Visitor,
 All of these patterns!

Program to interfaces,
 Not implementation, no, no, no!
 And favour composition
 over inheritance...oh!

We’re building our code for reuse,
 That we can plug and play,
 In many different ways,
 Design Patterns... in code.

ACKNOWLEDGMENT

The author wishes to thank Katie Cole for her arrangement of the music for the songs, for her conducting of the SWE Singers choir and for her feedback on the lyrics.

REFERENCES

- [1] G. Highet, *The art of teaching*. Taylor & Francis Group, 1951.
- [2] A. Ziv, “Teaching and learning with humor: Experiment and replication,” *The Journal of Experimental Education*, vol. 57, no. 1, pp. 4–15, 1988.
- [3] R. Schiffman. (2020, Oct.) Laughter may be effective medicine for these trying times. [Online]. Available: <https://www.nytimes.com>
- [4] J. Morreall, “Humor, philosophy and education,” *Educational Philosophy and Theory*, vol. 46, no. 2, pp. 120–131, 2014.
- [5] E. Hamilton, H. Cairns, L. Cooper *et al.*, *The collected dialogues of Plato*. Princeton University Press, 1961.
- [6] D. M. Martin, R. W. Preiss, B. M. Gayle, and M. Allen, “A meta-analytic assessment of the effect of humorous lectures on learning,” *Classroom communication and instructional processes: Advances through meta-analysis*, pp. 295–313, 2006.
- [7] R. A. Martin and T. Ford, *The psychology of humor: An integrative approach*. Academic press, 2018.
- [8] J. Bryant, P. W. Comisky, J. S. Crane, and D. Zillmann, “Relationship between college teachers’ use of humor in the classroom and students’ evaluations of their teachers,” *Journal of educational psychology*, vol. 72, no. 4, p. 511, 1980.
- [9] M. Machlev and N. J. Karlin, “The relationship between instructor use of different types of humor and student interest in course material,” *College Teaching*, vol. 65, no. 4, pp. 192–200, 2017.
- [10] R. G. Lomax and S. A. Moosavi, “Using humor to teach statistics: Must they be orthogonal?” *Understanding Statistics: Statistical Issues in Psychology, Education, and the Social Sciences*, vol. 1, no. 2, pp. 113–130, 2002.
- [11] C. C. Tisdell, “An arts-integrated approach to learning mathematics through music: A case study of the song “e is a magic number”,” *International Journal of Innovation in Science and Mathematics Education (formerly CAL-laborate International)*, vol. 27, no. 7, 2019.
- [12] L. Dao. (2016) 10 myths in software development to debunk now. [Online]. Available: <http://blog.enabled.com.au/myths-software-development/>
- [13] S. Adams, *The Dilbert principle: A cubicle’s-eye view of bosses, meetings, management fads and other workplace afflictions*. Pan Macmillan, 2000.
- [14] B. Meyer, *Agile!: The Good, the Hype and the Ugly*. Springer Science & Business Media, 2014.
- [15] N. Bekir, V. Cable, I. Hashimoto, and S. Katz, “Teaching engineering ethics: a new approach,” in *31st Annual frontiers in education conference. Impact on engineering and science education. Conference proceedings (Cat. No. 01CH37193)*, vol. 1. IEEE, 2001, pp. T2G–1.
- [16] E. Baniassad, “Making the liskov substitution principle happy and sad,” in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, 2018, pp. 17–20.
- [17] E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
- [18] H. Gardner. ((last accessed 11 October, 2020)) Home page. [Online]. Available: <http://users.cecs.anu.edu.au/Henry.Gardner/>
- [19] A. Field, *Discovering statistics using IBM SPSS statistics*. sage, 2013.
- [20] ——. ((last accessed 11 October 2020)) Discovering statistics home page. [Online]. Available: <https://www.discoveringstatistics.com/>
- [21] C. Larman, *Applying UML and patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development (3rd Edition)*. Pearson Professional, 2004.
- [22] P. Bourque, R. E. Fairley *et al.*, *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.