

# Petri Nets in Software Technology

## Scientific Writing II

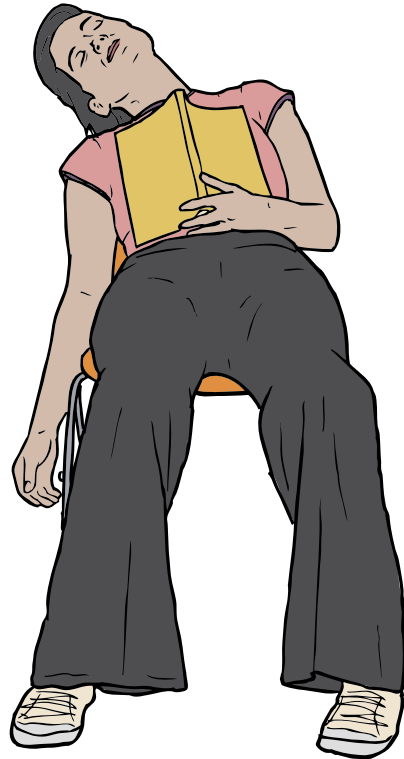
Hauptseminar (SS 19)

Wednesday, 2. DS, APB/3105

Thomas Kühn (thomas.kuehn3@tu-dresden.de)



## Reading



## Writing



## Organizing



Images from OpenClipart.org (Creative Commons by Steve Lambert)



## Common Tasks

- Deploy the given **template**
- Read and understand **author guidelines**
- Add all **meta information** (*title, authors, affiliations, ...*)
- Include your **abstract**
- **Structure** your document into subdocuments per chapter/section
- Include images, listings, and tables in **floating environment**

- Create your Camera Ready version with LaTeX<sup>1</sup>
  - Correct use the acmart template
 

```
\documentclass[sigconf,review]{acmart}
```
  - Add the copyright information
 

```
\setcopyright{rightsretained}
\acmConference[PTST'19]{Petri Nets in
Software Technology}{July 12}{Dresden,
Germany}
\acmYear{2019}
```
  - Add the correct acknowledgements of your funder
  - Check the *ACM SIGPLAN Conference Format*
    - Incorporate hints from the ACM website<sup>2</sup>
    - Follow the author guidelines



- 1) [www.latex-project.org](http://www.latex-project.org)
- 2) <http://www.sigplan.org/Resources/Author/>

### Comply to the Paper Geometry

- Fix **all** Overfull box errors<sup>3</sup>
  - Use a ruler or
  - `\usepackage[color=red, width=3pt, height=0.5\baselineskip]{overcolored}`
- Fix **some** Underfull box errors
- Check the boundaries of all images
- Set the correct geometry<sup>4</sup>
  - A4 → `a4paper`
  - US Letter → `letterpaper`
  - US Legal → `legalpaper`
  - ...

**Modular Feature-Oriented Graphical Editor Product Lines**

Thomas Kühn  
Software Technology Group  
Technische Universität Dresden  
Dresden, D-01062, Germany  
thomas.kuehn@tu-dresden.de

Kevin Ivo Kassin  
Software Technology Group  
Technische Universität Dresden  
Dresden, D-01062, Germany  
kevin\_ivo.kassin@mailbox.tu-dresden.de

Walter Cazzola  
Computer Science Department  
Università degli Studi di Milano  
Milan, I-20135, Italy  
cazzola@di.unimi.it

Uwe Almann  
Software Technology Group  
Technische Universität Dresden  
Dresden, D-01062, Germany  
uwe.assmann@tu-dresden.de

**ABSTRACT**

Software Product Lines (SPLs) have a long tradition and aim at reducing development costs by increasing reuse. They have been successfully applied to develop families of languages, ultimately establishing the field of Language Product Lines (LPLs). Currently, LPLs facilitate a family of textual languages by defining an SPL of complex interpreters. In contrast, this work aims at supporting families of graphical languages by defining an SPL of graphical editors, whereas each language variant is supported by a corresponding product of a Graphical Editor Product Line (GEPL). Thus, for these editors a modular approach for the development of GEPLs for families of visual languages. To remedy this, this paper introduces a feature-oriented development approach for GEPLs that ensures modularity, maintainability, and extensibility of the resulting product line. To showcase the usability and applicability of our approach, we developed a modular GEPL for the family of role-based modeling languages, a feature-rich family of conceptual modeling languages. Finally, we illustrate its extensibility by adding a complex language feature.

**KEYWORDS**

Software Product Lines, Language Product Lines, Modeling Languages, Graphical Editors, Product Lines

**ACM Reference format:**  
Thomas Kühn, Kevin Ivo Kassin, Walter Cazzola, and Uwe Almann. 2018. Modular Feature-Oriented Graphical Editor Product Lines. In *Proceedings of Software Product Line Conference, Gothenburg, Sweden, Sept. 30-14, 2018*. 11 pages.  
<https://doi.org/10.1145/3268888>

**1 INTRODUCTION**

The development of product lines has a long tradition and aims at reducing development costs by increasing reuse. Software Product Lines (SPLs), in particular, permit to drastically increase code reuse when creating product variants. In case of dynamic SPLs, this goes as far as to allow for dynamically changing product variants at runtime. Most recently, researchers successfully applied SPLs to develop families of languages, ultimately establishing the field of Language Product Lines (LPLs).

SPL '18, Sept. 30-14, Gothenburg, Sweden  
2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.  
<https://doi.org/10.1145/3268888>



18, Sept. 30-14, Gothenburg, Sweden

19 **Abstract**

20 **Keywords**

21 **ACM Reference format**

22 **1 INTRODUCTION**

23 The development of product lines has a long tradition and aims at

24 reducing development costs by increasing reuse. Software Product

25 Lines (SPLs), in particular, permit to drastically increase code reuse

26 when creating product variants. In case of dynamic SPLs, this goes

27 as far as to allow for dynamically changing product variants at

28 runtime. Most recently, researchers successfully applied SPLs to

29 develop families of languages, ultimately establishing the field of

30 Language Product Lines (LPLs).

31 SPL '18, Sept. 30-14, Gothenburg, Sweden

32 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

33 <https://doi.org/10.1145/3268888>

34 **1 INTRODUCTION**

35 The development of product lines has a long tradition and aims at

36 reducing development costs by increasing reuse. Software Product

37 Lines (SPLs), in particular, permit to drastically increase code reuse

38 when creating product variants. In case of dynamic SPLs, this goes

39 as far as to allow for dynamically changing product variants at

40 runtime. Most recently, researchers successfully applied SPLs to

41 develop families of languages, ultimately establishing the field of

42 Language Product Lines (LPLs).

43 SPL '18, Sept. 30-14, Gothenburg, Sweden

44 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

45 <https://doi.org/10.1145/3268888>

46 **1 INTRODUCTION**

47 The development of product lines has a long tradition and aims at

48 reducing development costs by increasing reuse. Software Product

49 Lines (SPLs), in particular, permit to drastically increase code reuse

50 when creating product variants. In case of dynamic SPLs, this goes

51 as far as to allow for dynamically changing product variants at

52 runtime. Most recently, researchers successfully applied SPLs to

53 develop families of languages, ultimately establishing the field of

54 Language Product Lines (LPLs).

55 SPL '18, Sept. 30-14, Gothenburg, Sweden

56 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

57 <https://doi.org/10.1145/3268888>

58 **1 INTRODUCTION**

59 The development of product lines has a long tradition and aims at

60 reducing development costs by increasing reuse. Software Product

61 Lines (SPLs), in particular, permit to drastically increase code reuse

62 when creating product variants. In case of dynamic SPLs, this goes

63 as far as to allow for dynamically changing product variants at

64 runtime. Most recently, researchers successfully applied SPLs to

65 develop families of languages, ultimately establishing the field of

66 Language Product Lines (LPLs).

67 SPL '18, Sept. 30-14, Gothenburg, Sweden

68 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

69 <https://doi.org/10.1145/3268888>

70 **1 INTRODUCTION**

71 The development of product lines has a long tradition and aims at

72 reducing development costs by increasing reuse. Software Product

73 Lines (SPLs), in particular, permit to drastically increase code reuse

74 when creating product variants. In case of dynamic SPLs, this goes

75 as far as to allow for dynamically changing product variants at

76 runtime. Most recently, researchers successfully applied SPLs to

77 develop families of languages, ultimately establishing the field of

78 Language Product Lines (LPLs).

79 SPL '18, Sept. 30-14, Gothenburg, Sweden

80 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

81 <https://doi.org/10.1145/3268888>

82 **1 INTRODUCTION**

83 The development of product lines has a long tradition and aims at

84 reducing development costs by increasing reuse. Software Product

85 Lines (SPLs), in particular, permit to drastically increase code reuse

86 when creating product variants. In case of dynamic SPLs, this goes

87 as far as to allow for dynamically changing product variants at

88 runtime. Most recently, researchers successfully applied SPLs to

89 develop families of languages, ultimately establishing the field of

90 Language Product Lines (LPLs).

91 SPL '18, Sept. 30-14, Gothenburg, Sweden

92 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

93 <https://doi.org/10.1145/3268888>

94 **1 INTRODUCTION**

95 The development of product lines has a long tradition and aims at

96 reducing development costs by increasing reuse. Software Product

97 Lines (SPLs), in particular, permit to drastically increase code reuse

98 when creating product variants. In case of dynamic SPLs, this goes

99 as far as to allow for dynamically changing product variants at

100 runtime. Most recently, researchers successfully applied SPLs to

101 develop families of languages, ultimately establishing the field of

102 Language Product Lines (LPLs).

103 SPL '18, Sept. 30-14, Gothenburg, Sweden

104 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

105 <https://doi.org/10.1145/3268888>

106 **1 INTRODUCTION**

107 The development of product lines has a long tradition and aims at

108 reducing development costs by increasing reuse. Software Product

109 Lines (SPLs), in particular, permit to drastically increase code reuse

110 when creating product variants. In case of dynamic SPLs, this goes

111 as far as to allow for dynamically changing product variants at

112 runtime. Most recently, researchers successfully applied SPLs to

113 develop families of languages, ultimately establishing the field of

114 Language Product Lines (LPLs).

115 SPL '18, Sept. 30-14, Gothenburg, Sweden

116 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

117 <https://doi.org/10.1145/3268888>

118 **1 INTRODUCTION**

119 The development of product lines has a long tradition and aims at

120 reducing development costs by increasing reuse. Software Product

121 Lines (SPLs), in particular, permit to drastically increase code reuse

122 when creating product variants. In case of dynamic SPLs, this goes

123 as far as to allow for dynamically changing product variants at

124 runtime. Most recently, researchers successfully applied SPLs to

125 develop families of languages, ultimately establishing the field of

126 Language Product Lines (LPLs).

127 SPL '18, Sept. 30-14, Gothenburg, Sweden

128 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

129 <https://doi.org/10.1145/3268888>

130 **1 INTRODUCTION**

131 The development of product lines has a long tradition and aims at

132 reducing development costs by increasing reuse. Software Product

133 Lines (SPLs), in particular, permit to drastically increase code reuse

134 when creating product variants. In case of dynamic SPLs, this goes

135 as far as to allow for dynamically changing product variants at

136 runtime. Most recently, researchers successfully applied SPLs to

137 develop families of languages, ultimately establishing the field of

138 Language Product Lines (LPLs).

139 SPL '18, Sept. 30-14, Gothenburg, Sweden

140 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

141 <https://doi.org/10.1145/3268888>

142 **1 INTRODUCTION**

143 The development of product lines has a long tradition and aims at

144 reducing development costs by increasing reuse. Software Product

145 Lines (SPLs), in particular, permit to drastically increase code reuse

146 when creating product variants. In case of dynamic SPLs, this goes

147 as far as to allow for dynamically changing product variants at

148 runtime. Most recently, researchers successfully applied SPLs to

149 develop families of languages, ultimately establishing the field of

150 Language Product Lines (LPLs).

151 SPL '18, Sept. 30-14, Gothenburg, Sweden

152 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

153 <https://doi.org/10.1145/3268888>

154 **1 INTRODUCTION**

155 The development of product lines has a long tradition and aims at

156 reducing development costs by increasing reuse. Software Product

157 Lines (SPLs), in particular, permit to drastically increase code reuse

158 when creating product variants. In case of dynamic SPLs, this goes

159 as far as to allow for dynamically changing product variants at

160 runtime. Most recently, researchers successfully applied SPLs to

161 develop families of languages, ultimately establishing the field of

162 Language Product Lines (LPLs).

163 SPL '18, Sept. 30-14, Gothenburg, Sweden

164 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

165 <https://doi.org/10.1145/3268888>

166 **1 INTRODUCTION**

167 The development of product lines has a long tradition and aims at

168 reducing development costs by increasing reuse. Software Product

169 Lines (SPLs), in particular, permit to drastically increase code reuse

170 when creating product variants. In case of dynamic SPLs, this goes

171 as far as to allow for dynamically changing product variants at

172 runtime. Most recently, researchers successfully applied SPLs to

173 develop families of languages, ultimately establishing the field of

174 Language Product Lines (LPLs).

175 SPL '18, Sept. 30-14, Gothenburg, Sweden

176 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

177 <https://doi.org/10.1145/3268888>

178 **1 INTRODUCTION**

179 The development of product lines has a long tradition and aims at

180 reducing development costs by increasing reuse. Software Product

181 Lines (SPLs), in particular, permit to drastically increase code reuse

182 when creating product variants. In case of dynamic SPLs, this goes

183 as far as to allow for dynamically changing product variants at

184 runtime. Most recently, researchers successfully applied SPLs to

185 develop families of languages, ultimately establishing the field of

186 Language Product Lines (LPLs).

187 SPL '18, Sept. 30-14, Gothenburg, Sweden

188 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

189 <https://doi.org/10.1145/3268888>

190 **1 INTRODUCTION**

191 The development of product lines has a long tradition and aims at

192 reducing development costs by increasing reuse. Software Product

193 Lines (SPLs), in particular, permit to drastically increase code reuse

194 when creating product variants. In case of dynamic SPLs, this goes

195 as far as to allow for dynamically changing product variants at

196 runtime. Most recently, researchers successfully applied SPLs to

197 develop families of languages, ultimately establishing the field of

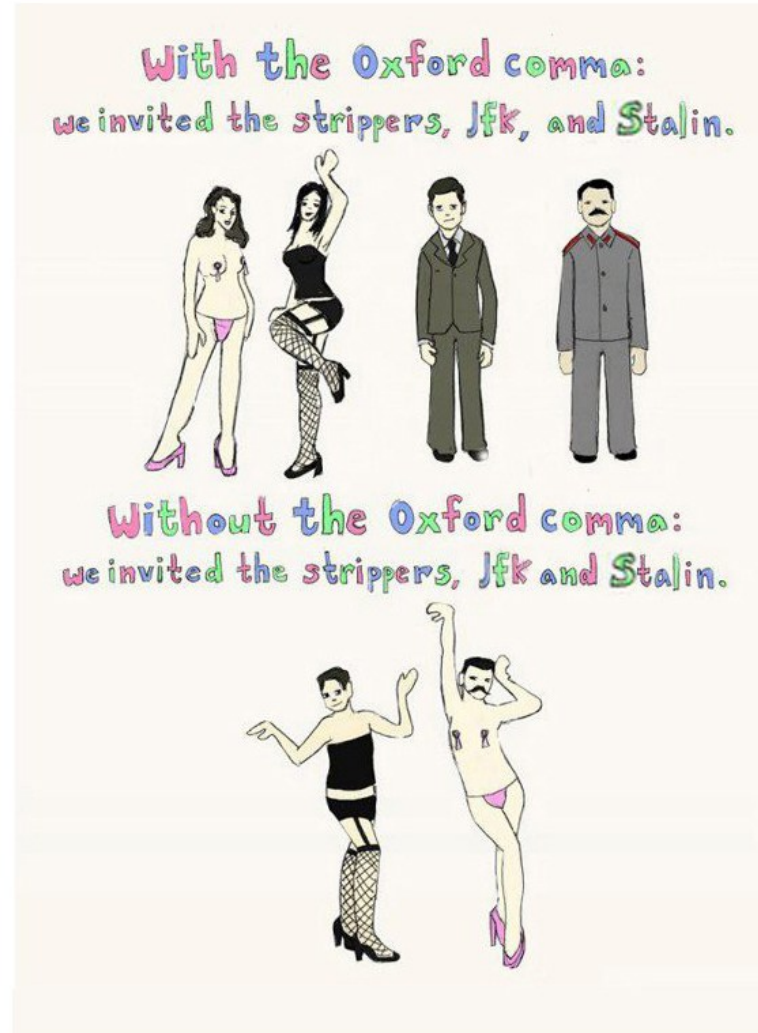
198 Language Product Lines (LPLs).

199 SPL '18, Sept. 30-14, Gothenburg, Sweden

200 2018, ACM, ISBN 978-1-4503-5932-7, 11 pages.

201 <https://doi.org/10.1145/3268888>

- 3) <https://github.com/raphink/overcolored>
- 4) <https://ctan.org/pkg/geometry>



- Check for spelling and grammar mistakes
  - Use a spell checker,<sup>5</sup>
  - Revise your own text, and
  - Employ a proofreader
- Check punctuation
  - Comma before accessory sentence with *because, which, ...*
  - No comma before *that*
  - Comma after *therefore, nonetheless, in conclusion, ...*
  - Use the oxford comma<sup>6</sup>

5) <https://www.youtube.com/watch?v=8Gv0H-vPoDc>

6) <http://richardgilbert.me/we-like-the-oxford-comma-ii>

## Chicago Manual of Style (CMS) title capitalization rules<sup>7,8</sup>

1. Capitalize first and last words in titles and subtitles (but cf. 7), and capitalize all other major words (*nouns, pronouns, verbs, adjectives, adverbs, and some conjunctions—but see rule 4*).
2. Lowercase the articles *the, a, and an*.
3. Lowercase prepositions, regardless of length, except when used adverbially or adjectivally (*up in Look Up, down in Turn Down, on in The On Button, to in Come To, etc.*) or when they compose part of a Latin expression used adjectivally or adverbially (*De Facto, In Vitro, etc.*).
4. Lowercase common coordinating conjunctions *and, but, for, or, and nor*.
5. Lowercase not only as a preposition (rule 3), but also as part of an infinitive (*to Run, to Hide, etc.*), and lowercase as in any grammatical function.
6. Lowercase the part of a proper name that would be lowercased in text, such as *de* or *von*.
7. Lowercase the second part of a species name, such as *fulvescens* in *Acipenser fulvescens*, even if it is the last word in a title or subtitle.

7) <http://www.chicagomanualofstyle.org/book/ed17/part2/ch08/psec159.html>

8) <http://www.chicagomanualofstyle.org/book/ed17/part2/ch08/psec161.html>

**Usual**

alternative  
comparable  
complement  
dependent  
descendant  
discrete  
emit  
ensure  
ensure  
excerpt

**Other**

alternate  
comparative  
compliment  
dependant  
descendent  
discreet  
omit  
insure  
assure  
exert

**Usual**

foregoing  
further  
elusive  
manyfold  
omit  
partly  
principle  
simple  
solvable  
stationary

**Other**

forgoing  
farther  
illusive  
manifold  
emit  
partially  
principal  
simplistic  
soluble  
stationery



**Right**

adaptation

apparent

argument

comparison

consistent

definite

existence

foreign

grammar

heterogeneous

homogeneous

independent

insoluble

**Wrong**

adaption

apparant

arguement

comparision

consistant

definate

existance

foriegn

grammer

heterogenous

homogenous

independant

insolvable

**Right**

miniature

occasional

occurred

participate

primitive

propagate

pronunciation

pseudo

referred

repository

separate

supersede

transparent

**Wrong**

minature

occaisional

occured

particepate

primative

propogate

pronounciation

psuedo

refered

repositery

seperate

supercede

transparant

- Include a non-breaking space before each citation  
modeling languages~\cite{authorE}
- Citations at end of a sentence  
“conducted, as well. [5]” → “conducted [5], as well.”  
“modeling languages. [27]” → “modeling languages [27].”
- Correctly list citations  
“[1], [2], [3], [6]” should be “[1–2,6]”  
\cite{authorA,authorB,authorC,authorF}

- Fix footnotes occurring before punctuations

“family of RMLs<sup>3</sup>.” → “family of RMLs.<sup>3</sup>”

“feature<sup>12</sup>,” → “feature,<sup>12</sup>”

“major drawback <sup>123</sup>” → “major drawback<sup>1,2,3</sup>”

```
\usepackage[multiple]{footmisc}
```

- Footnotes with links utilize `\url{https://...}`

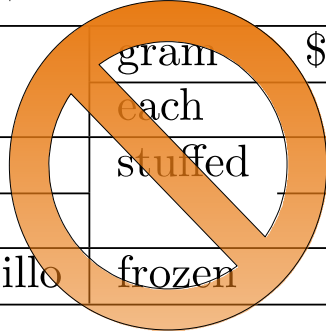
```
\usepackage{url}
```

- Footnotes in paper title and authors with `\titlenote` and `\authornote`

- (Almost) all figures, listings, and tables
  - Floating environment
  - Align top or all on page
- ACM guidelines for captions
  - For figures **below**
  - For tables **above**
  - For listings **above**
- Guidelines for tables
  - Use booktabs for tables
 

```
\usepackage{booktabs}
```
  - **Only** use horizontal lines, **never** vertical lines
 

```
\toprule \midrule \bottomrule
```
  - **Never** use double line



|           |         |         |
|-----------|---------|---------|
| gnats     | gram    | \$13.65 |
|           | each    | .01     |
| gnu       | stuffed | 92.50   |
| emu       |         | 33.33   |
| armadillo | frozen  | 8.99    |

| Item      |             |            |
|-----------|-------------|------------|
| Animal    | Description | Price (\$) |
| Gnat      | per gram    | 13.65      |
|           | each        | 0.01       |
| Gnu       | stuffed     | 92.50      |
| Emu       | stuffed     | 33.33      |
| Armadillo | frozen      | 8.99       |

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Fusce nec pharetra justo.  
Nam in augue quis dolor

---

**bibendum gravida eget elit.**

**Lorem ipsum dolor sit amet,**

---

consectetur adipiscing elit.  
Fusce nec pharetra justo.  
Nam in augue quis dolor  
bibendum gravida eget elit.

Fusce vestibulum finibus nisi  
id fermentum. Nullam cursus  
urna enim, ac feugiat nunc  
sollicitudin vel integer feugiat  
**tincidunt.**

### Widow

- Last sentence of paragraph on new page

### Orphan

- First sentence of paragraph on previous page

### Orphan (alternative)

- Last word of paragraph alone on last line

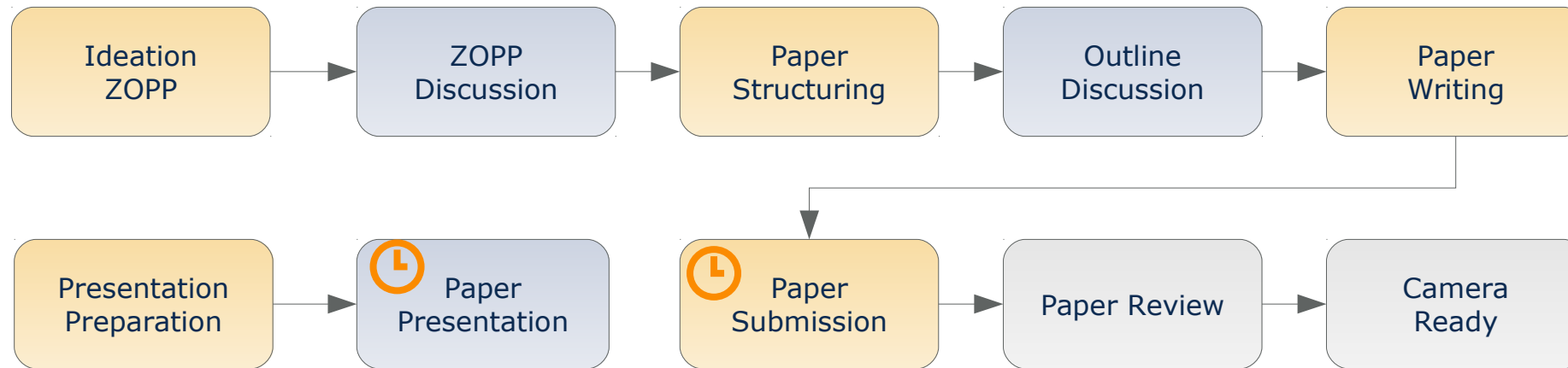
9) [https://en.wikipedia.org/wiki/Widows\\_and\\_orphans](https://en.wikipedia.org/wiki/Widows_and_orphans)

- Add missing information to bibliography
  - Every paper needs at least
    - Title, authors, year, publisher, **page numbers**
    - Conference or journal
  - Optional, but useful: Digital Object Identifier (DOI)
  - Optional, but waste of space: Uniform Resource Locator (URL)  
(Only in case of *websites, white papers, thesis, and videos*)
- Balance the last page of references (in case of two-column styles)  
`\usepackage{balance}`

- Prepare the PDF to be uploaded
- Create archive with the camera ready source code (usually \*.zip)
  - Create a separate folder
  - Include all used *latex source* (\*.tex)
    - Remove spacing trickery ( $\ll[-0.5em]$ )
    - Remove all comments
  - Include all used code *snippets, images, and external artifacts*
  - Include the (\*.bbl) generated bibliography file  
(*instead of the full bibliography (\*.bib)*)
- *Commit and tag* the version of your camera ready sources

## Submission

- Upload PDF document and archive
- Sometimes document is checked for spacing errors



- *Give scientific presentations* (20 min + 10 min discussion)
  - Individual presentations **05.06.2019**
- *Write a research paper* ( $\geq 5$  pages ACM Style)
  - Paper submission<sup>1</sup> **12.07.2019**

1) Per mail



# Software for Digital Health

## Scientific Writing

