What Is on the Horizon?

We have discussed various aspects of meta-programming throughout the book. The paradigm contributes to automatic generation of programs. Linguistic aspects, which describe how programming languages or domain-specific languages are to be used within the paradigm, are at the focus. As languages differ in their capabilities (e.g. computational models and modes of use), a very broad spectrum of the language-centric generative approaches (e.g. generative programming, generic programming, aspect-oriented programming, feature-based programming) can be viewed as meta-programming now. Our approach differs from the mentioned ones at least by two aspects. First, our approach manipulates with programs or their parts using a meta-language in the mode of structural programming. Second, we use a variety of languages as meta-languages (from dedicated to general-purpose programming ones). The dedicated meta-language we have developed implements the concept of external functions. The functions can be seen as an extended list of pre-processing commands independent upon any target language.

On the other hand, when dealing with meta-programming, we can accept different views to the paradigm, such as those discussed within the book: (1) meta-programming as a program generalization and (2) meta-programming as a model-driven process in developing meta-programs. Though both visions are not mutually exclusive, the first approach focuses more on linguistic aspects and transformations within language environments, while the second approach moves transformations towards higher levels of abstraction and focuses more on higher-level models such as feature models. Thus, the model- and transformation-based view is important because such a vision extends the boundaries for dealing with and understanding meta-programming in the large. In that aspect, we fully agree with the Czarnecki and Helsen hierarchy of terms: meta-programming – model transformation – program transformation, where meta-programming is at the top and encompasses the meaning of the remaining terms.

Now, we can raise two general questions: (1) What is above meta-programming in the vertical dimension? (2) What is the way to extend meta-programming in the horizontal dimension? The latter question entails a traditional view, that is,
language-based paradigms to meta-programming and transformations. This trend encompasses all the above-mentioned language-centric approaches. Though each of them is further evolving, it seems (according to the number of scientific publications and industrial support) that aspect-oriented programming has received a more attention and effort. A feature-based programming is the only recently adopted concept, and it needs more research. The expansion of meta-programming in the horizontal dimension conforms that the paradigm is seen as a technology for the automatic program construction. But the automatic program construction is not the primary goal. The primary goal is the creating of a system as effectively as possible. In that aspect, meta-programming can be seen as a technology enabling to separate design tasks from programming tasks due to a great deal of anticipation, which is achievable through the increased extent of analysis and variability modelling.

Let us return to the first question. It relates to the product line engineering (PLE) approach in the following sense. The PLE focuses on the feature variant analysis and variability modelling; thus, it supports the anticipation in designing a family of related systems. Those activities are treated as domain engineering now. At the very high abstraction level, domain engineering separates the design and programming activities. Below domain engineering, there are activities known as application engineering. In general, all kinds of meta-programming aim at connecting domain engineering with application engineering. The efforts to raise the abstraction level (as a response to complexity challenges) lead also to meta-engineering. The latter includes meta-analysis, meta-modelling and meta-design. Those meta-activities are yet not well-understood and require a great deal of research. They are driving forces to move research in meta-programming in the vertical dimension.

In the context of the discussed concepts, we envisage the research trends for further generalization and extension the meta-programming topic in the following directions:

- Further integration of knowledge from related research domains (e.g. program and model understanding, feature-based modelling, program and model changeability) to extend model-driven generalization of heterogeneous programming.
- Expanding of the role of domain-specific meta-programming, where meta-programming per se can be seen as a domain-specific one, to achieve a higher degree of reuse and automation.
- Expanding of the role of meta-programming in terms of variability and changeability anticipation for software maintenance and evolution.
- Meta-meta-modelling and meta-meta-programming or generative meta-programming in the context of multi-linguistic and domain-specific programming approaches.
- Model-driven transformations of feature-based models to develop meta-programs in a variety of domain-specific applications.
- Contextualization of feature-based models, inventing of feature-based patterns for the specification of meta-programming tasks and the adequate transformation rules development.
Further generalization of meta-programming can be also researched in terms of theoretical approaches, such as partial evaluation and multi-stage programming.

In a broader context, meta-programming can evolve as a technology to support meta-design and meta-engineering.

More accurate complexity evaluation estimates are also seen on horizon.

Expansion of application domains, where feature-based modelling and meta-based programming could be the relevant implementing technologies.

Formalization of transformation-based meta-programming is also seen as an agenda for research.

The formulated predictions are motivated by the following observations:

1. Analysis of general trends in advancement of technology learned from the literature during writing of the book, readings and research work
2. Analysis of specific-literature and related topics cited within the book, where such signs are already noticeable at a different degree of their population and maturity
3. Personal experience and intensive research work in the field accomplished by both authors in the field during the last decade.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>Application Engineering</td>
</tr>
<tr>
<td>CFD</td>
<td>Contextual Feature Diagram</td>
</tr>
<tr>
<td>CI</td>
<td>Cyclomatic Index</td>
</tr>
<tr>
<td>DA</td>
<td>Domain Analysis</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DE</td>
<td>Domain Engineering</td>
</tr>
<tr>
<td>DL</td>
<td>Domain Language</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain-Specific Language</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>FD</td>
<td>Feature Diagram</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FODA</td>
<td>Feature-Oriented Domain Analysis</td>
</tr>
<tr>
<td>FT</td>
<td>Forward Transformation</td>
</tr>
<tr>
<td>GLO</td>
<td>Generative Learning Object</td>
</tr>
<tr>
<td>GPL</td>
<td>General-Purpose Programming Language</td>
</tr>
<tr>
<td>He MPG</td>
<td>Heterogeneous meta-programming</td>
</tr>
<tr>
<td>HL</td>
<td>High-Level</td>
</tr>
<tr>
<td>Ho MPG</td>
<td>Homogeneous meta-programming</td>
</tr>
<tr>
<td>KC</td>
<td>Kolmogorov Complexity</td>
</tr>
<tr>
<td>LL</td>
<td>Low-Level</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-Up Table</td>
</tr>
<tr>
<td>MB</td>
<td>Meta-Body</td>
</tr>
<tr>
<td>MDA</td>
<td>Model-Driven Architecture</td>
</tr>
<tr>
<td>MDD</td>
<td>Model-Driven Development</td>
</tr>
<tr>
<td>MDSoC</td>
<td>Multi-Dimensional Separation of Concepts</td>
</tr>
<tr>
<td>MI</td>
<td>Meta-Interface</td>
</tr>
<tr>
<td>ML</td>
<td>Meta-Language</td>
</tr>
<tr>
<td>MMP</td>
<td>Meta-meta-program</td>
</tr>
<tr>
<td>MP</td>
<td>Meta-program</td>
</tr>
<tr>
<td>MPG</td>
<td>Meta-programming</td>
</tr>
<tr>
<td>Abbr.</td>
<td>Definition</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>MTT</td>
<td>Model Transformation Tool</td>
</tr>
<tr>
<td>Open PROMOL</td>
<td>Open for extension PROgram MODification Language</td>
</tr>
<tr>
<td>PLE</td>
<td>Product Line Engineering</td>
</tr>
<tr>
<td>RKC</td>
<td>Relative Kolmogorov Complexity</td>
</tr>
<tr>
<td>RT</td>
<td>Reverse Transformation</td>
</tr>
<tr>
<td>TL</td>
<td>Target Language</td>
</tr>
<tr>
<td>VHDL</td>
<td>VLSIC – Very Large Scale Integrated Circuit Hardware Description Language</td>
</tr>
<tr>
<td>VS</td>
<td>Variation Sequence</td>
</tr>
<tr>
<td>VSM</td>
<td>Variation Sequence Modelling</td>
</tr>
</tbody>
</table>
Index

A
Abstraction
  high abstraction level, 115, 131, 132, 172, 185, 291
  higher-level of abstraction, 7, 20, 21, 93, 217, 218, 313
  level of abstraction, 7, 20–22, 25, 176, 217, 219, 232, 293, 307
  lower-level of abstraction, 20

C
Complexity
  algorithmic complexity, 212, 223
  cognitive complexity, 210, 212–214, 221, 231, 232, 234
  cyclomatic complexity (CC), 212, 234
  design complexity, 81, 210, 211, 213, 214, 233, 243
  feature model complexity, 233, 234
  Kolmogorov complexity, 217, 218, 234
  meta-program complexity, 210, 223–225, 227–230, 233, 235
  program complexity, 222, 233, 261
  system complexity, 13, 211, 212

Component
  component family, 19, 218–220, 229
  component generator, 280, 284, 286, 288, 309
  generative component, 19, 24, 109, 123, 243, 295, 296, 301
  generic component, vii, 6, 19–22, 24–26, 67, 73, 83, 86–88, 93, 107, 255, 293
  HW component, 293
  meta-component, 19, 172, 173, 178, 179, 202, 203, 285, 302, 313, 314
  meta-meta-component, 202, 302, 303, 314
  web component, 6, 280–284, 286, 287, 312

Context
  context-based feature, 297
  context-based modelling, 203, 297
  contextual feature diagram, 297, 314

D
Domain
  domain abstraction, 24, 130, 132–134, 137
  domain commonality, 84, 119, 165
  domain engineering, 82, 119, 173
  domain generator, 171, 281, 284, 286, 311, 312, 314
  domain language, 220, 254
Domain (cont.)
domain modelling, 80, 84, 196, 203
domain variability, 80, 81, 83, 119, 120, 123, 126, 129, 135, 165, 166, 198–200, 222, 248
well-understood domain, 80, 84, 109, 313

F
Feature
alternative feature, 44, 146, 181, 183, 196
feature-based model, vi, 43, 119, 125, 129, 166, 173, 179, 185, 198, 203, 234, 291, 297
feature constraints, 301
feature relationship, 132, 164, 198, 215
feature selection, 180, 183, 301
feature variant, 43, 123, 124
grouped features, 180
mandatory feature, 44, 146, 152, 165, 181, 297
optional feature, 44, 146, 181, 183
solitary feature, 44, 146
variation sequence, 153–163, 166

G
Generative technologies
aspect-oriented programming, 6, 34, 50, 106, 157, 211, 212, 241, 254, 255
feature-based programming, 317
generative programming, 6, 34, 209, 241, 254
generic programming, 6, 49, 58, 211, 212, 241, 254, 255, 305
Generic
generic component, vii, 6, 19–22, 24–27, 67, 73, 83, 86–88, 93, 107, 255, 293
generic functionality, 83, 88, 94
generic interface, 58, 65, 69–71, 83, 88, 314
generic programming, 6, 49, 58, 211, 212, 241, 254, 255, 305

I
Information hiding, 10, 31, 33, 34, 43, 45, 191, 209, 214

L
Language
domain language, 20, 22, 23, 27, 36, 37, 84, 85, 89, 90, 93, 122, 124, 135–139, 191, 196, 197, 199, 202, 205, 220–223, 225, 232, 279, 281, 293, 294, 296, 306, 314
domain-specific language, v, 27, 50, 60, 66, 67, 74, 94, 177, 246
functional language, 97, 108
general purpose programming language, 66, 67, 73, 109, 247
language environment, 85, 317
Perl, 245, 247
scripting language, 90, 94, 95, 97, 105, 108
systemC, 94, 183, 203–205, 246, 295, 308
target language, 36, 49, 60, 93, 96, 97, 99, 108, 192, 200, 242, 245, 246, 256, 261, 286, 287, 311
**M**

**Meta-meta-program**
- meta-design, 6, 10, 11, 27, 89, 115, 118, 120–126, 132, 134, 135, 140, 244, 248, 259
- meta-generator, 10, 192, 207, 244, 311, 313
- meta-meta-program model, 257, 260

**Meta-program**
- meta-construct, 18, 90, 135, 189, 205, 261, 263, 267
- meta-program complexity, 210, 223–225, 227–230, 233, 235
- meta-program complexity measures, 210
- meta-program design, 38, 129, 140, 189, 201, 248
- meta-program lifecycle, 117, 247
- meta-program model, 136, 137, 141, 189, 200, 257, 260, 282
- meta-program transformation, 267, 268, 274

**Meta-programming**
- heterogeneous meta-programming, v, vi, 8, 11, 12, 23, 37, 39, 46, 79–81, 83, 85, 86, 88, 90, 91, 94, 104, 107, 109, 115, 116, 125, 128, 221, 241, 242, 245, 247, 248, 250, 257, 279, 284, 293, 311
- meta-programming techniques, 6, 31, 52, 59–64, 73, 80, 123, 221, 231, 280, 281, 284, 287, 302, 313
- multi-stage meta-programming, 202, 242, 243
- structural heterogeneous meta-programming, v, vi, 79, 91, 191, 243

**Model**
- abstract model, 88, 141
- formal model, 18, 129, 234, 255
- meta-meta-model, 11, 133
- meta-modelling, 6, 43, 84, 115, 130, 132, 133
- model-driven analysis, 140
- model representation, 132, 140, 152, 257
- model transformation, 80, 128, 130–132, 141, 190, 200, 205–207, 249, 260, 271, 286
- Monad, 60, 61, 64, 65, 74

**P**

**Parameter**
- external parameterization, 122
- high-level parameterization, 38, 45, 248
- parameter space, 230, 258–260, 262, 267, 268, 273, 274
- parameter value, 39, 42, 69, 70, 72, 83, 100, 103, 105, 121, 162, 197–200, 204, 224, 230, 259, 262, 264, 306
- set of parameters, 88, 259

**Program**
- program change, 109, 120, 173, 178, 185, 186, 228, 248, 254, 255
- program changeability, 173, 178, 185, 186, 254, 255
- program family, 96, 157
- program generalization, 177, 178, 241
- program manipulation, 5, 12, 32, 36
- program modification, 95, 109
- program specialization, 268
- program transformation, 6, 18, 79, 130, 131, 189, 253–256, 267, 268, 274

**R**

**Reflection, 6, 22–25, 105, 106, 124**

**S**

**Separation of concepts**
- explicit separation of concepts, 35
- implicit separation of concepts, 35, 68
- multi-dimensional separation of concepts, 39, 41, 45, 46

**Specification**
- meta-program specification, 38, 43, 116, 118, 122, 125, 139, 191, 244
- multi-language specification, 27
Taxonomy
taxonomy of meta-programming concepts, v, 23
Template, 6, 24, 25, 52, 53, 60, 62–64, 71–74, 144, 222, 227, 231, 232, 255, 256, 284
Transformation
forward transformation process, 272–274
reverse transformation process, 179, 274
semantics-preserving transformation, 267, 269, 274
transformation task, 179, 189, 206, 242, 273
two-stage reverse transformation, 271, 273