5.4 Data Science, Artificial Intelligence, and Machine Learning

One of the themes of early work in artificial intelligence (AI) was referred to as "automatic programming". The essential diversity and complexity of software and the underlying tacit and difficult to precisely express technical and domain-specific knowledge required to build industrial-quality software were beyond the feasibility of then-known techniques. This shortcoming was the motivation for defining the analogous capabilities of descriptive metaprogramming for building and using adaptable software and of prescriptive metaprogramming for transformation, optimization, and instrumentation of derived content [see section 4.1].

Developers in building products have for sometime relied on code completion features and repositories of previously developed content. Recent AI-related advances, leveraging the experience represented in such retained content, have begun to support automated derivation of customized software content. As with the use of adaptable components for deriving customized content, the DsE methodology, being method-agnostic, accommodates use of these advances in deriving customized product model content.

{product family concepts can also be a basis for characterizing a set of assets that can be generatively derived—abstractions as descriptors of similarity is meaningful for understanding a generative capability—domain definition as basis for characterizing the limits of needed content}

These AI-related advances entail data science and machine learning techniques applied in concert with AI reasoning techniques to realize the concept of "generative" AI[1]. Data science techniques provide the means to analyze, categorize, and profile previously developed software assets; AI techniques provide the means to standardize and discriminate among similar problem-solution alternatives based on behavioral (functional and quality) criteria; and various machine learning techniques provide the means to extrapolate from prior problem-solution experiences to more quickly and reliably generate content for new products.

[1]"What is generative AI?", IBM, 2024. <https://www.ibm.com/think/topics/generative-ai>; "Glossary of LLM Terms", Vectara, 2024. < https://www.vectara.com/glossary-of-llm-terms>

[The focus here is exclusively on potential improvements in product development capabilities that may be feasible with emerging AI-based techniques. Enhanced customer-directed product capabilities that may be enabled with such techniques are not considered further except as being also relevant for product development. Such capabilities include endeavors such as action planning, analysis of alternatives, prediction of outcomes, management of complex scenarios, obtaining and applying expert knowledge in a subject area, coordinating a complex information space, etc.]

Foundations and Limitations

Past AI approaches for knowledge-based inferential reasoning (aka expert systems) provided a medium for applying specialized human expertise in domain-specific solutions. The concept of adaptable software was conceived as a practicable analogous form for applying this approach in software development. Newly developed techniques, based on the concepts of data analytics and neural networks, focus on statistical analyses of large repositories of legacy software content to discover similarities among instances that provide the basis for deriving similar content that is then combined with other content to solve a specified problem. Current work will continue, along lines discussed below, toward deriving and comparatively evaluating alternative content against quality criteria and identifying assemblies of derived content that are more complete solutions to more complex problems.

Several concerns need to be considered in determining whether these techniques will be effective in improving software practice; participants in generative AI development are actively pursing resolutions to these concerns:

{need more refinement of the following items}

- Will a generative AI trained in a specific area of subject matter provide better (faster, more reliable) responses to questions in that area? (i.e., is domain competence an advantage?) Can a general AI focus on authoritative sources of subject matter to ignore extraneous information?

- Can the cost (including time and energy use) of machine learning (analyzing source materials to generate neural net models) be reduced? Does a narrowed focus on particular subject matter material be of benefit?
- Can generative AI be built to recognize limits in its scope and depth of competence so as to decline questions that exceed its competence?
- Is an inability to produce usable results due to insufficient relevant subject matter source material recognized? Are known unknowns, omissions, and uncertainties in source content able to be identified and mitigated in results?
- Can generative AI provide rationale, explaining its reasoning and identifying sources used, for purposes of evaluating the validity of its results?
- Is there be an ongoing process for continuously revising and verifying the competence of a generative AI as relevant source information changes? Is there an associated process for correcting incomplete/biased or erroneous results?
- Is there a process for identifying and including tacit or otherwise omitted information (e.g., assumed but not explicitly stated information on which a product is based)?
- Is there a means for recognizing and correcting factual versus reasoning errors in generative AI? Can "facts" be rated as to belief and accuracy to be considered in reasoning? Is there a mechanism for identifying and resolving missing, incomplete, or conflicting information?
- Can the same AI distinguish and be used for both factual and fictional or conjectural reasoning (i.e., without including invalid content in factual reasoning)?
- How are the sources and quality of content determined as sufficiently representative of potential product scope and having verifiable integrity (e.g., provenance, attribution, currency)? Are results routinely updated as relevant source material changes (created, updated, or corrected)?

- Is there an ongoing process of verifying that results are stable/predictable when reproducing results in analyzing unchanged content (i.e., consistency of results given a prescribed level of probabilistic variation)?

Given the relatively limited practical experience with AI-based techniques, answers to these concerns may already be emerging or could be entirely beyond the means of current techniques, requiring the conception of other not yet known techniques. Nevertheless, the following discusses capabilities that, from the perspective of the DsE methodology, would improve the practice of software development.

Topics for Generative AI

Generative AI may provide opportunities for advances in software development capabilities in a product family context that would otherwise be infeasible or require substantial effort to achieve. These advances relate to understanding product behavior, optimizing product quality factors, automating product derivation, and exploiting leverageable similarities in products.

Profiling Product Behavior

These opportunities concern gaining confidence that a product upon deployment will reliably exhibit intended behavior. The objective is to provide understanding of a product that is either time-consuming or infeasible to obtain otherwise.

- Ensured consistency: Expose inconsistencies in content of product model elements, particularly specification-realization pairs, for deriving a sound product
- Explicated product behavior: Derive explanations of requirements-projected or observed product behavior (normal, defect, anomaly), supporting product evaluation reviews or for user understanding
- Automated product verification: Given informal descriptions or histories of operational use, generate scenarios to create and apply a test suite with data and inputs→expected outputs, noting differences in results from expected

- Unspecified behavior: Discover potential (observed or inferred) behavior, traced to product content, that has not been specified in product requirements

Diagnosing Product Quality

These opportunities concern how quality factors of varying importance combine to achieve a product having an envisioned level of overall quality. These focus on how alternative versions of a product can differ in how they address quality, how different weightings of quality factors affect overall quality, and which components most significantly influence the various quality factors. The objective of these is to explore options for optimizing quality before deployment and without the effort required to obtain empirical results.

- Component-level quality: Determine the contribution of a component to satisfying relevant quality factors, for use in evaluating how changes will affect those factors and aggregate quality
- Decision-implied quality: Derive a product quality profile for alternative resolutions of critical requirements or engineering decisions to determine how resulting product versions will differ in behavioral quality.
- Capability-quality tradeoffs: Evaluate whether modifying requirements or engineering decisions (e.g., possibly limiting customer expected product capabilities), will resolve unmet quality criteria.

Generating Product Content

These opportunities concern options for streamlining the derivation of a product. These focus on deriving components of each work product and deriving complete products, including alternative versions either for comparison or for alternative uses.

- Singular software components: Generate one or more implementing modules, including design (interface and internal specifications), implementation, and verification elements, from an informal text description, optionally referencing similar existing content

- Adaptable components: Generate an adaptable component (software, document, or test suite) based on specified parameters of variation with means to derive an instance component that fits resolved parameters
- Interface-alignment: Generate an implementation partitioned into a set of interface-consistent components
- User documentation: Generate customer documentation and training materials consistent with product requirements & environment model specifications
- Virtual ecosystem: Generate a virtual ecosystem from an informal description (text/diagram/image) and product environment model specification of its information content and entity elements, optionally with specified behavior

Leveraging Similarity

These opportunities concern an ability to characterize an envisioned set of products that are sufficiently similar to satisfy a specified abstraction. These focus on the premise that differences can be expressed as deferred decisions that in being resolved are sufficient to generate a customized instance of the envisioned set.

- Example-based assumptions: Derive a hierarchically organized set of commonality and variability assumptions that characterize the similarities in a set of previously built products
- Coalesced products: Generate a representation of a product family, based on a (partial) set of notionally similar existing products, from which those and other similar products can be derived
- Deferred decision effects: Summarize the product subfamily that is determined by a given resolution of a partial set of deferred decisions (e.g., commonalities that distinguish the subfamily, subsequent decisions that remain unresolved)
- Similarity metric: Derive a similarity measure (i.e., distance metric) for the metric space defined by a product family as a basis for finding behaviorally “close” alternatives to an instance (e.g., based on how differing decision model resolutions correspond to differences in behavior); associate decision resolution

differences with the transitional link between each pair of closely related products [see Appendix, section 5]

- Extended formal methods: Generalize applicable formal methods that, applied to a product family, will identity which instances will satisfy specified quality factor criteria [see section 5.2]
- Product manufacturing capability: Construct a generalized product manufacturing capability that, with the specification of a domain model, can be used to generate product models