DiverSE Coffee

Retrospective 10 years
Triskell 2012

- Research Program (RA 2012)
  - Model Driven Engineering for Distributed Software
  - Software Product Lines
  - Object-Oriented Software Engineering
  - Design Pattern
  - Component
  - Contracts
  - Models and Aspects
  - Design and Aspect Weaving
  - Model Driven Engineering

- Software Language Engineering (SLE)
  Separation of concerns in the development of complex software-intensive systems leads to the use of various domain-specific modeling languages (DSMLs). SLE addresses the whole life cycle for designing, implementing and relating DSMLs to support heterogeneous modeling and analysis.

- Variability
  The systematic modeling of variability in software systems has emerged as an effective approach to document and reason about software evolutions and heterogeneity. Variability modeling characterizes an "envelope" of possible software variations.

- Dynamic Adaptive System (DAS)
  Flexible yet dependable systems have to cope with heterogeneous hardware execution platforms ranging from smart sensors to huge computation infrastructures and data centers. We need to devise formalisms to reason about the impact of an evolution and about the transition from one configuration to another.

- Diversity
  Software diversity as the foundation for a novel software design principle and increased adaptive capacities in complex adaptive systems. Higher levels of diversity in the system provide a pool of software solutions that can eventually be used to adapt to unforeseen situations at design time.
Programmez! Numéro #150 (01/03/2012)
– Dossier “Fier d’être développeur!”
Aute exemple de titres des années 2010...
Vision en 2012
Capitalisation du savoir faire inter disciplines

• Inter actions entre experts de différents domaines

• Vers une capitalisation du savoir faire
  – Bonnes pratique et process de développement
    • ALM (Application Lifecycle Management)
  – IDM/MDE permet d’aller plus loin en permettant
    • une manipulation à la fois par les machines et par les humains
    • Réapplication de techniques inventées pour un domaine à de nouveaux domaines
Utilisation d’abstractions

- Les développeurs sont habitués au changements de niveau d’abstraction pour gérer une complexité croissante
  - Assembleur > language procéduraux > language orientés objets + framework > langages orientés modèles / DSL
  - Utilisation de l’abstraction adaptée au besoin + utilisation de compilateurs
    - Périodes de transitions pour pallier aux limites des compilateurs
Separation of concerns

- Convergence modèle/code
  - Syntaxes textuelles / graphiques
- Découplage des préoccupations
  - Techniques orientés aspects
    - Au niveau code (via des fonctionnalités de langage telles que scala, Ruby C#, ou via injection par des annotations)
    - Et généralisées à toutes les étapes de conception

Figure 1 : Tissage semi automatique des aspects et génération de code
Importance des processus

● Humain dans la boucle
  — Paramétrage, choix des outils, itérations successives sur les modèles
  — Gestion du cycle de développement (BPMN/SPERM/…)
  — Adaptabilité permettant la mise en œuvre de ligne de produit logicielle
Vers plus de fiabilité

- support aux systèmes dynamiques
- Adaptation @DesignTime (grâce au MDE dans les phases de conceptions)
- Adaptation @Runtime
  - Reconfiguration à chaud

Figure 2 : Reconfiguration dynamique des fonctions d'un téléphone pour s'adapter à une nouvelle situation.
Conclusion de 2012

• Technique permettant de répondre aux enjeux de:
  − Accélération des technologies
    • Multiplication des architectures
  − Besoins hybrides
  − Complexité croissante / spécialisation croissante
  − Maîtrise du processus dans son ensemble
    • Modularité des outils
Et aujourd’hui ?

Sujet “modelling” dans cette revue

• Les termes IDM/MDE n’apparaissent plus récemment dans cette revue généraliste

• Mais on trouve
  - LowCode/NoCode
    • #260 (03/11/2023)
    • #255 (02/12/2022)
    • Hors série #7 (05/2022)
    • #244 (30/10/2020)
  - Domain specific
    • Thread Modeling #255
• Quels points vous semblent-ils toujours d’actualité ?
  – Éventuellement sous une autre terminologie ?
• Quels points semblent en perte de vitesse ?
• Avenir d’un écosystème MDE dans l’open source ?
  – Sensibilité des écosystèmes de dev
“My” Vision 2024

- better integration/collaboration of the all activities (incl. DevOps)

- Tomorrow developers tools
  - Concern aware collaborative workspaces
    - Or “Concern aware social workspaces”?
  - Human is the center, the tool(s) should help keeping the focus while providing info about events in its environment/dependencies

- Content of what is dreamed next already exists in a myriad of tools, let’s imagine all this assembled and rationalized
Some problems to address

- Difficulty to add the tooling
  - User is not expert
- Need to provide the right level of details at a given time
  - About the system being built/analysed
  - About the process
- Analysis / monitoring / debug
- Verification / validation
- Prediction / simulation
- Decision making / adaptation
  - Early feedbacks
- Adaptive run location
- Adaptive run precision
- Increase interest in extra functional aspects
- Use modern methods at all levels
  - Clever use of resources

Problems to address

- Many stakeholders
- Many points of view
- Usually provide a single point of view (technology of project centric)
  - Should allow focus on the user
  - Should allow focus on the project
  - Should allow focus on a specific task
  - Process is not explicit
  - History explainability
  - Missing global management views
  - Interoperability issues
    - Polyglot coordination
    - Technologies
    - Languages
    - Working environments
    - Quality of the tool issues
  - Traceability matters
  - Global management issues
  - Missing global management views
  - Domain specific (name your domains here)
  - Technology specific (name your technology here)
  - Security
  - Safety
  - Maintainability
  - Economy
  - Vulnerability
  - Accessibility (RGAA)
  - Sustainability
  - Law
  - Documentation
  - Regulation (RGPD)
  - License
  - Quality
  - Ethics
  - ...
Concern aware collaborative workspaces requirements

- Concern/profile aware
  - Focus on some activities
  - Other information shown using more abstract/compact representations (zoom)
    - (all?) other concerns should still be present without requiring to switch to the concern dedicated workspace (kind of “Gemini cricket” showing impact on/of other concerns)
      - User/project should be able to customize level of details / threshold for warning / frequency of reminders
        - Easily switch to dedicated workspace if focus changes
  - Non exhaustive list of concerns (as main or insight):
    - Domain specific (name your domains here), Technology specific (name your techno here), security, safety, Economy, vulnerability, accessibility (RGAA), sustainability, regulation (RGPD), law, License, Documentations (tutorials, how to guides, explanation, reference), ...
- Some concerns should appear both about the build process and the system under construction/analysis
  - Ex: sustainability, vulnerability, economy,
Concern aware collaborative workspaces requirements

• Verification support
  — “Are we developing the software application/model correctly?”
  - Should include all static checks (and proof)
    • Automated locally, remotely
    • Not automated (done by the user or any collaborator)
  - Should apply to both the application and the build system
    • Ex: detection of supply chain vulnerability
Concern aware collaborative workspaces requirements

• Validation support
  – “Are we developing the **right** software application/model?”
  – Should include support for all executions
    • Test suite executions
    • Local executions
    • Production execution (monitoring and debug of the prod)
Concern aware collaborative workspaces requirements

• Natively polyglot
  – Works on both code AND models
    • ie. know that a given concept can be represented using different formalisms
  – With debug support
    • Omniscient (ie. forward and backward)
  – With for forecast
    • Prediction
    • Live programming
Concern aware collaborative workspaces
requirements

• Adaptable
  – Adapt fidelity of related models/codes
    • from mock to high fidelity to real/physical system
  – Adapt tool execution location
    • Move any tool or execution between User computer and remote resources (such as CI/CD runners, LLM queries, verification...)
Concern aware collaborative workspaces

requirements

• Traceable

  - Every “significant” action should have a mean to be related its origin
    • Ex: see a debug break point state in its different facets (models, codes, assembler)
    • Ex: human activities are saved and/archived (kind of version control) (on demand / automatically)
Concern aware collaborative workspaces

requirements

• Explainable/reproducible
  – List of tool used (dependencies, configuration, version, ...)
    • (also required to be able move to remote computing resource
  – Store evolutions of sources AND ( store meaningful executions results OR a way to reproduce results on demand a given version)
    • Results include functional data, but also non functional data such as execution time
    • For regression detection
Concern aware collaborative workspaces

requirements

• Natively Social
  - Should allow human in the loop for any task of the workflow
  - Should allow pair programming
Concern aware collaborative workspaces

requirements

• Explicitable overall process
  – if not customized by the user, the tool must be able to present the selected process
Concern aware collaborative workspaces requirements

• LLM integration
  – Workflow should define the validation process of LLM inputs
  – Integration using Human interface
    • Ex: create pull request (ex: dependabot)
    • similar to human, activity and result should be traced
    • Ex: chat
    • Ex: assign task or issues similarly to a human but to a bot
  – Integration as tool
    • Ex: wizard, refactor, translation, transformation, ...
    • As any tool of the process, usage and result should be traced
  – traceability in case of auto validation without human control
Concern aware collaborative workspaces

requirements

• Open
  – Plugin...
How to achieve that?

- We have several work in the team that can contribute
- Opportunity to build something larger than prototypes?
- Which base technology?
  - Risk if too specific: reproduce Eclipse twilight?
- Many tools to integrate how to smoothly integrate them?
- How to avoid a big bloatware?
  - Create collaborating subsets? (eg. forge de l’ingénieur + ?)
- Suggestion:
  - Massively invest on protocols?
    - BSP, DAP, LSP, ...
    - Improve protocol specification for better interoperability (ie. REST vs SocketRPC vs GraphQL vs gRPC vs tRPC vs Protocol Buffers…) or find bridges to avoid the protocol hell about protocol type, transport protocol, request/response format, schema language, support of subscriptions, …
Autres sujets du dossier de 2012

- Mixité
  - 2012
    • environnement très masculin
  - 2024
    • Et aujourd’hui pensez vous que l’on ai progressé ?

- Développeur
  - 2012
    • mal vu d’être développeur toute sa vie. (peu valorisant)
    • Il faut être chef de projet, architecte.
    • La France reste toujours en retard sur d’autres pays comme l’Angleterre ou les Etats-Unis où être développeur n’est pas une tare, et même le contraire
  - 2024
    • Est ce toujours le cas ?