24

VC life cycle : phases

25

Roles in VCs

- Platform operator*: omnipotent role
- Initiator
  - Create, Set the community parameters (topics, duration, dynamics…). Owner.
- Leader*
  - Modify the community description (shapes the topic), delete, monitor and moderate content
  - Motivates members, glue members together
- Member
  - Read, produce, declare profile, download content
  - Different levels of expertise from novice to expert… (may lead to different rights
- Visitor (possibly): Limited rights

*A crucial role

26

Users and communications

- Many different possibilities
  - Operator to platform members
  - Operator to community leader
  - Leader to member
  - Leader to community
  - Member to leader
  - Member to member
  - Member to community
  - Leader/member to visitor , etc…
- 1 to 1, 1 to n
- Possibly n to 1, n to n (vote, petition)

27

Actions in the different phases

28

VC platform actions (1/3)

- **Initiation functions**
  - Set community parameters: topics, vocabulary, etc.), dynamics (protocol for recruitment, production, data analysis, etc.)
  - Create community. Ex: Group together people listening to the same music artists on a streaming platform
- **Recruitment & Profile functions**
  - Promote a community to users (platform members or external)
  - Select users as potential candidates (send invitations, or inform the leader)
  - Update user's profile (based on behavior and content)

29

VC platform actions (2/3)

- **Production and *moderation* functions**
  - Store contents
  - Policy enforcement
    - Detect unappropriated contents
    - Execute process engine
  - Calculations, analysis
    - Distances: user-user, user-content, content-content,
    - Contents' evolutions (ex: folksonomy)
  - Search engine, display contents, users' info
    - Index content
    - Promote recommended content, triggered alerts
  - Deliver reward, punishment (banish users)
  - Provide extra services: voting system, calendar management, workflow engine (reminders, incentives)

30

VC platform actions (3/3)

- ***Fork function***
  - Detect sub-communities (evaluation of the need), Inform actors
  - Launch community initiation function (manage information storage)
- **Sleep/Terminate**
  - Detect community inaction, manage information deletion/backup

31

What about Mobile Communities?

Transversal characteristics

- Use of mobile devices
  - Smart phone
  - Wearable sensors
- Connection during the mobility
- Emphasize the (evolving) context of the users: time, space, social environment. Impact on the user's profile.

32

Examples of Community platforms

- Task centered
- User centered
- Topic centered

33

Task centric communities
Collaborative ontology development

- Ontology (Gruber93) : formal specification of terms and relations in a domain
- Useful for organizing and categorizing data (used on Internet : Web of Data micro-format)
- Ontology development steps
  - Specify the domain
  - Initiate a list of terms of the domain, classes (hierarchy), relations between classes
  - Search for existing ontologies
  - Iterate on the list of terms till group members are satisfied
  - Populate the ontology (= create a knowledge base)

34

Collaborative ontology development

- Why collaborative?
  - Ontologies represent a consensus on a domain
  - Different expertises: various points of views, domain experts, end-users, knowledge engineer
- List of requirements for collaborative ontology development
  - Proposed by Todorache 2008
  - Extended by Al Qawasmeh 2018
  - This list is useful to compare different methods/tools

35

Collaborative ontology development
Requirements 1/3

- R1 **Discussion**: sharing ideas, comments, queries
- R2 Support of various level of expressiveness: from high-level description to very **detailed** formalization of the domain
- R3 **User management and Provenance information**: users should be identified, actions should be traced
- R4 **Scalability, reliability, robustness**: ability of the tool to support big ontologies; and to offer trusted stable environment

36

Collaborative ontology development Requirements 2/3

- R5 **Access control**: Read/write access on parts of the ontology granted to users
- R6 **Workflow support**: Definition and execution of ordering rules for tasks (example: edition step → validation step)
- R7 **Access type**: synchronous or asynchronous

37

Collaborative ontology development Requirements 2/3

- R8 **User friendly**: easy to use for all kinds of users (not familiar with such tools)
- R9 **Tool's setup, open source code**: easiness of installing the tool; possibility of control the tool's content, and to extend the functionalities
- R10 **Ontology validation**: checking functions such as syntax and semantic validation
- R11 **Supported format**: existing and produced ontology may be in different formats: owl, rdf, turtle…

38

Collaborative ontology development Tool: Protégé

- From Stanford University http://protege.stanford.edu/
- Most famous tool for ontology edition
- Free, open source (Java), organized as a framework with plugins
- Features: graphic user interface, consistency validation, new information inferencing
- WebProtégé for online tool
  - Limited functions : multi-users, comments/discussion attached to objects http://webprotege.stanford.edu

39

Collaborative ontology development Tool: VoCol

- From University of Bonn and Fraunhofer IAIS
- Features (integration of several components):
  - **Versioning control system, based on github for tacking of evolutions**
  - Text editor with syntax validation (turtle) (easy to use for domain experts)
  - Checking of inconsistency/constrains
  - Editor free (theoretically!) (format translation for a unique serialization service)
  - Documentation generation (HTML), visualization
  - Query service (through a SPARQL end point)
- Collaborative features fully rely on Github
- Difficult to setup; No freedom in the generated output; Bugs or missing functions (it is not a product)

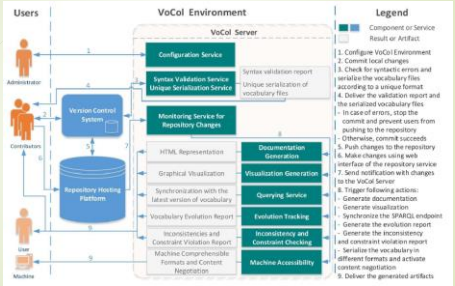
40

Principles of Github

- Dedicated to controlling versions of projects
- User identify into repositories (repo)
- Repository: contains all files related to a project
- Main concepts in the main page
  - **Code:** structure & contents of the repo
  - **Issues:** discussion topics, problems related to the repo
  - **Pull Requests:** list of changes for review / acceptance
  - + **Commits:** all saved and documented changes to the repo
  - + **Fork**: personal copy of the repo → **Pull request**: transfer the changes to initial repo
  - + **Pull** to create local (offline) copy, **Push** to send changes to the forked repo

41

Vocal Architecture and workflow



42

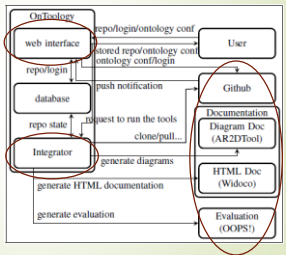
Collaborative ontology development Tool: Ontoology

- From University Polytechnic of Madrid (Alobaid 2015)
- Based on Github (similarly to Vocol)
- External tools : Documentation, Visualization, Evaluation
- Role of Ontoology (see architecture)
  - User's access to Github repositories; Setup of the ontologies (tools activated)
  - Launch tools: Production of diagrams, HTML documentation, evaluation report
  - Communicate with Github : submit documents (pull request) to be reviewed/merged by the maintainer ; Add issues for the pitfalls generated by the evaluation tool

43

Ontoology architecture

- Web interface
- Integrator
- External components
  - Github
  - Documentation service (Widoco)
  - Visualization (AR2DTool)
  - Evaluation (OOPS!)



44

Evaluation of Task centric communities/ Collaborative ontology development

- **In-use experiment and comparison**
  - Difficulties: availabilities of experts, subjectivity of modeling, importance of the user interface
- **Compare tools based on a list of requirements**
  - Izquierdo 2016 : Analyse distributed ontology engineering tools through agile approach (continuous development, quick responses to change, collaborative)
  - See next slide

45

Comparison table of collaborative ontology development tools (from Alqawasmeh)

	Category 1					Category 2			Category 3		
	R1	R2	R5	R10	R11	R3	R6	R7	R8	R9	Comments
Domingue [10]	✓	✓	✓	NI	NI	✓	NI	both	NI	✗	–
Arpirez et al. [11]	✓	✓	✓	NI	NI	✓	✓	NI	NI	✗	Arch. for ontology dev.
Kozaki et al. [12]	✗	✓	✓	NI	NI	✓	NI	NI	NI	✗	–
Sure et al. [13]	✓	NI	NI	✓	✓	✓	✓	NI	NI	NI	–
Hayes et al. [14]	✗	✓	✓	NI	NI	NI	NI	NI	NI	✗	–
Komulainen et al. [16]	NI	✓	✓	✓	✓	✓	✓	NI	NI	NI	Frame work
Auer et al. [17]	✓	✓	✓	✓	✓	✓	✓	NI	NI	NI	–
De Leenheer and Debruyne [18]	✓	✓	✓	✗	NI	✓	✓	NI	NI	NI	–
Noy et al. [19]	✗	✓	✗	✓	✓	✓	✓	Sync	✓	✓	–
Tudorache et al. [21]	✓	✓	✗	✓	✓	✓	✓	Sync	✓	✓	–
Ghidini et al. [22]	NI	✓	✓	NI	NI	✓	NI	NI	NI	NI	–
Tawfik et al. [23]	✓	NI	✓	NI	NI	✓	NI	Async	NI	NI	–
Alobaid et al. [24]	✗	✓	✓	✓	✗	✓	✗	NI	✓	✓	based on Github
Halilaj et al. [25]	✗	✓	✓	✓	✗	✓	✗	Async	✓	✓	based on Github

R1: Discussions, R2: Various levels of expressiveness, R3: Management and provenance of information
R5: Access control, R6: Workflow support, R7: Access type, R8: User Friendly
R9: Easy to Install/Configure- open source, R10: Ontology validation, R11: Supported ontology types
Y: YES, N: No, NI: No Information

Vocal

Ontology