Collaboration in open source software development: the ALERT approach

Ljiljana Stojanovic, Jun Ma, Sinan Sen
Business process management (BPM) is a systematic approach to making an organization's workflow more effective, more efficient and more capable of adapting to an ever-changing environment.

There are two kinds of business processes:

- Mechanistic business processes are on the most part implemented by machines and human involvement is limited to key decision and data entry points.
  - They are routinized and often semi- or fully automated.

- Human-driven processes, differ from this in that they are fundamentally collaborative, dynamic and innovative.
  - These processes depend on interaction and are dynamically shaped by the participants.
Agenda

- Introduction
- Our approach
- Conceptual architecture
- Conclusion
Introduction

- Large scale software development is an inherently collaborative, team based process, and hence requires **coordination** and control in order to make it successful.
- The use of **communication** technologies (e.g., e-mails) alleviates problems caused by separation of workers in time and space, studies have often found them to be not as effective as publicized.
- The main problem is that, in order to enhance coordination, communication must be **efficient** (i.e. timely and clear) and **effective** (i.e. targeted).
FLOSS communities use a **variety** of communication tools. This makes the information pull very **heterogeneous, distributed and not well-linked** on the basis of its content
- mailing lists for technical discussions, a BTS for monitoring and fixing bugs, a CVS code repository for storing a common version of the code, etc.

FLOSS communities are very **communication-intensive**, which causes a kind of **information overload**
- The KDE Bugzilla contains 18000/15500 open bugs/ feature. During the last year 38000 Bugzilla entries were opened
- There are about 100 active KDE mailing lists which accumulate about 1400 messages a day

FLOSS developers are said to **autonomously** decide how and when to contribute to project development. This makes the **awareness** about the work and need of others very vague
Agenda

- Introduction
- **Our approach**
- Conceptual architecture
- Conclusion
Bob asks about Nepomuk
Mike confirms Mike reports on forum
Dirk asks Andreas how to fix bug
Andreas suggests Mike’s fix

John reports a bug
Bob asks for explanation
John replies …
Mike links to 253312
Dirk asks for explanation
Dirk applies Mike’s fix

1st Aug 25th Aug 1st Sep 12th Sep 16th Sep 20th Sep 24th Sep 28th Sep 3rd Oct 7th Oct 22nd Oct

Disc 1
Disc 2
IRC
Research challenges

- How to coordinate and maintain awareness of community activities?

The information and interactions relating to the bug were located in multiple artefacts with no links. Thus, a duplicate bug was filed.
How to interpret (understand) communication?

There is a lack of coherence that hinders developers in making sense of communications from others, or that produces unintended information filtering or misunderstandings.
Research challenges

-How to identify expertise?
The bug was assigned to/noticed first by someone without expertise in the area of the bug
How will the ALERT system address the identified challenges?

Interpretation

Awareness

Task Assignment
How will the ALERT system address the identified challenges?
How will the ALERT system address the identified challenges?

Information source 1

Information source 2

I.s. 3

I.s. n

continuous push of changes

INTERPRET

INTERACTION HIGHWAY

CREATE AWARENESS

AUTOMATIC ASSIGNMENT

Developer 1

Developer 2

Developer m

complex subscriptions

preferred information
Alert System

Summary: Dolphin Qt-dbus random crashes (Nepomuk integration?)

Product: dolphin

Reporter: Dusan Zirojevic
<duzan.zirojevic@cimcollege.rs>

Assignee: Peter Penz
<peter.penz19@gmail.com>

Component: general

Status: UNCONFIRMED

Severity: normal

CC: aleksandar.siskovic@cimcollege.rs, ivan.ohradovic@cimcollege.rs

Priority: NOR

Version: unspecified

Target Milestone: ---

Hardware: Unlisted

OS: Binaries

Version Fixed In: Linux

Resolution From Dusan Zirojevic 2010-10-05 16:20:18

unspecified (using KDE 4.3.1)

Base System release 2.0.0. -- Backtrace: Application: Dolphin (crash)

signal: Segmentation fault [Current thread is 0 (LWP 8630)]

Did you try to exit the application?
ALERT SYSTEM

Real-time knowledge extraction & integration
ALERT SYSTEM

Summary: Dolphin Qt-dbus random crashes (Nepomuk integration?)

Product: dolphin
Reporter: Dusan Zirojevic 
<duzan.zirojevic@cimcollege.rs>

Component: general
Assignee: Peter Penz
<peter.penz19@gmail.com>

Status: UNCONFIRMED

Severity: normal

CC: aleksandar.siskovic@cimcollege.rs

Version Fixed In:

Resolution From Dusan Zirojevic 2010-10-05 16:20:18

unspecified (using KDE 4.3.1)

Base System release 2.0.0" -- Backtrace: Application: Dolphin (unmn), signal: Segmentation fault [Current thread is 0 (LWP 8630)]

Possible: Didn't try
ALERT SYSTEM

Intelligent services
Ontologies as a data model
Ontologies as a data model

Models semantically interlinked online communities
Models persons
Models the competency of the developer

Models events, event types and event sources
Models BTS, ITS, source code, wikis, blogs, documentation, etc.

Models project related terms

Community Layer
- SIOC Ontology
- Competence Ontology

Interaction Layer
- Models project related terms
- Information Source Ontology
- Annotation Ontology
OBSEERVE
Event capturing
Interaction ontology

ORIENT
Event interpretation
Content and
Community ontologies

DECIDE
Complex event processing
Interaction patterns

ACT
Interaction patterns

- An interaction pattern is the description of the situation which should be detected and reported in real-time (immediately after it has appeared).
- Interaction patterns represent knowledge about the reactive behavior of the system (when a system should react on – from any reason).
- Main features:
  - Real-time: Get the information immediately.
  - Personalized: Get only the relevant information.

Example: Inform me whenever there is a new bug followed by an intensive discussion in the forum (e.g. intensive = more than 10 posts in five hours).
Interaction patterns

- There are different complexity tiers of interaction patterns that can be detected. They correspond to the complexity of the situations they describe:
  - **Tier 1** *low complexity*: interaction patterns can describe a situation that a property value is above/below a specific threshold, or that a specific event occurred
  - **Tier 2** *medium complexity*: Interaction patterns can describe a situation that extends **Tier 1** with the capability of combining multiple events from different event sources
  - **Tier 3** *high complexity*: Interaction patterns can describe a situation that extends **Tier 2** with the capability of windows (temporal, spatial, count) and aggregations
## Examples of interaction patterns

| Tier 1 | If there is a new bug with certain [properties] |
|        | If there is a new bug from [user] |
|        | If there is a new bug containing [words] |

| Tier 2 | If there is a commit and the bug in the package is not closed in BTS |
|        | If there is a new release and a bug related to this release is not closed in BTS |
|        | If there is a new [email],[wikiPage],[post] related to [method],[class],[package],[bug] |
|        | If there is a bug and there is a discussion about this bug that the bug has been fixed (because of a release) and bug status in BTS is not marked as fixed |

| Tier 3 | If there is new bug and nothing else happens related to this bug in all sources within a time period [T] |
|        | If there is a new bug and nothing else happens with this bug in BTS and there is a hot conversation in [blog],[wiki],[email],[forum] within a certain time period [X] |
|        | If there is a new bug without description followed by a [post],[wiki] containing a description for this bug |
Agenda

- Introduction
- Our approach
- Conceptual architecture
- Conclusion
- Using single and multiple criteria recommendation methods:
  * Recommend most competent developer to resolve an issue
  * Recommend a set of issues that are most suited to the developer's competence patterns
<event xmlns="http://alert-project.com/" xmlns:onto="http://alert-project.com/ontologie">
  <!-- common information about an event -->
  <head>
    <!-- string value -->
    <timestamp />
    <!-- by which alert component -->
    <sentby />
  </head>

  <!-- put the related information into the payload section -->
  <payload>
    <onto>
      <!-- property name="hasTime" value="2009-09-30" typeOf="http://www.w3.org/TR/xmlschema-2/#string" -->
      <!-- property name="doneBy" typeOf="http://#Person" -->
      <class value="http://personX"/>
      <!-- property name="hasName" value="Zsombor" typeOf="http://www.w3.org/TR/xmlschema-2/#string" -->
      <onto:property></onto:property>
    </onto>
    <onto:property></onto:property>
    <onto:property></onto:property>
  </payload>
</event>

<eventType name="NewBugEvent" typeOf="http://icep.fzi.de/eventrepresentation1.0/icepEventScheme#NewBugEvent">
  <onto:property name="isRelatedTo" typeOf="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bug">
    <class value="http://Bug1"/>
    <!-- property name="hasID" value="1234" typeOf="String" -->
    <onto:property name="hasStatus" typeOf="http://#BTSSStatus">
      <class value="http://status1"/>
      <!-- property name="hasDescription" value="RESOLVED" typeOf="http://www.w3.org/TR/xmlschema-2/#string" -->
      <onto:property></onto:property>
    </onto:property>
    <!-- property name="isAbout" typeOf="http://#Integration" -->
    <class value="http://TopicO"/>
    <!-- property name="hasImportance" value="2" typeOf="http://www.w3.org/TR/xmlschema-2/#integer" -->
    <onto:property name="hasDescription" value="Nepomuk" typeOf="http://www.w3.org/TR/xmlschema-2/#string"/>
    <!-- property name="isRelatedSourceCode" typeOf="http://#Method" -->
    <onto:property></onto:property>
  </onto:property>
</eventType>
Agenda

- Introduction
- Our approach
- Conceptual architecture
- Conclusion
Conclusion

- ALERT system will improve **FLOSS coordination** by maintaining awareness of community activities through real-time, personalized, context-aware notification.

- The ALERT system will act as an **active collaboration platform**, i.e. a virtual actor will
  - **interact with other** developers,
  - process and recognize various kinds of interactions,
  - suggest actions on the basis of these and remember and
  - bring past interactions into the developers’ attention, thus enabling developers to work better together.
Thank you!