October 29, Bern

Dear editor, dear reviewers.

Thank you for taking the time to review the article and the tool. We appreciate a lot your work and feedback.

We took into consideration your observations and are submitting an improved article together with this letter. The main contributions of this new version are split into two categories that we summarize here:

Tool related:

- We have created a better installation mechanism for Softwarenaut. For Windows and MacOSX you do not need to obtain and install VisualWorks before you can use the tool.
- We have created a better documentation for the system which can be accessed online at http://scg.unibe.ch/softwarenaut/book

Article Related:

- We have introduced a new section which discusses the evolution-related aspects of the tool. This was a requirement of several of the reviewers and we think it actually improved the article considerably.
- We have separated also the work on Collaboration into its own section and enriched a little bit that part too.
- We rewrote the section on the hierarchical graph model of Softwarenaut to make it clearer.
- We have improved what used to be section 6.1 and now is 7.2: the section which describes our usability study. We did not have enough time to run a new study, but we provided more detailed information about the study and the feedback we received from the participants.
- To make the article read better we introduced as a running case-study ArgoUML on which we illustrate the various features of the tool.

Finally, since his contributions in improving the article we have invited Oscar Nierstrasz to be co-author.

We attach to this letter the detailed answers for the observations of each of the reviewers and thank you again for your feedback.

Best regards,
Mircea Lungu, Michele Lanza, and Oscar Nierstrasz.

Reviewer #1: The paper

The paper describes Softwarenaut, an architecture recovery and visualization tool. The paper is **well structured** and **clearly explains the motivation** behind the development of the tool as well as its main features.
The application of the tool, as the authors outline it, is "the semi-automated discovery of architectural views of any object-oriented system and allows the sharing of such architectural views." The application areas are reengineering and software evolution monitoring. I think that the variety of the presented tool?s features suites to such application areas. The **objective - the architecture recovery of large software systems- is also clearly stated**.

Sections 2 to 4 describe how the objective of the tool is achieved. The features and the implementations details are, in general, explained quite clearly. The only unclear detail is whether or not such features of object oriented languages as asynchronous invocations (e.g., event model and dependency injection) can be reflected by the tool. I understand that the tool depends on the information given by fact extractors but, still, perhaps such cases of indirect method invocation are worth mentioning. The same holds for pointer invocations (like callbacks).

A: The tool is extensible with new types of dependencies, as long as a fact extractor can be written that would analyze the source code and export those dependencies in the MSE format. At the moment, the fact extractors only export information about static method calls although there is reasearch in parsing J2E specific deployment descriptors. We have added a clarification about this in the text.

Some small remarks:
p. 4 line 43: the method names do not correspond to the figure 2.
p. 5 line 34: Inconsistent with Fig. 3 (class may be a leaf).
p. 7 bottom to p.8: Do we really need formalization here? The operations are intuitively clear to the reader. In any case, the formalization is incomplete: "+?" and "-"operations are not defined. The names in the definitions do not correspond completely to the ones in the where clauses (What?s N? HG?). How do you define "virtual module"?

A: Fixed. We have removed the formalization.

Anyway, the tool’s set of features described in the paper is quite impressive. The visualization, although quite traditional (boxes and arrows), looks nevertheless modern enough.

A: To increase the “modern look” we have added a new screenshot which has the modules represented with a TreeMap-like representation.

In section 5 the authors explain that the tool’s input format is FAMIX. It?s also clear that the ability of the tool to represent software system accurately and completely depends heavily on the availability of a fact extractor producing data in such a format. The authors
also mention the tool"s affiliation to the Moose platform. Both, FAMIX and MOOSE, have quite a long history. Although the technology and the format itself are not very widely accepted, they are still very much alive and the community is active. The pluses are extensibility, language independence and the fact that the platform is free. However, as I mentioned above, the tool depends on FAMIX (MSE) parsers, which are currently available for a relatively small variety of languages.

Neither can find you here an explanation of design and implementation decisions. Why, e.g. Smalltalk? Moose and CodeCrawler, for instance, are implemented as standalone applications. This way you could?ve got rid of the dependency on Cincom Smaltalk.

A: Unfortunately, at the moment the system runs inside Cincom Smalltalk (and so does CodeCrawler and the versions of Moose before it’s porting to Pharo Smalltalk) and we can’t change that. The plans for the future are indeed to move from Cincom Smalltalk to Pharo Smalltalk, mainly for licence reasons.

However, in order to address the observation and “get rid” of the dependency to Cincom Smalltalk, we have prepared a stand-alone executable also for the Windows platform.

Section 6 is devoted to the tool building and evaluation considerations. In my opinion, this section is weak. You cannot see here any useful fro other developers reflections on the issues encountered during the design and building process.

The evaluation part is not convincing enough. You would expect a somewhat deeper statistical analysis here, objective and subjective comparison with- and without the tool, etc. If such data is not yet available, then **perhaps this section should be skipped and postponed to future work**.

A: We have rewritten the evaluation part. We renamed it to “Studying the Usability of the Tool” so it does not promise more than it is. We have made the text more rigorous. Unfortunately the time was not sufficient to run a more detailed experiment. If the reviewer still thinks that the section is not sufficient we can remove it.

The language of the paper is sufficient.

A: We have also seen worse :)

The tool.

That was a major disappointment for me. After reading the paper I was looking forward to put my hands on it.

The tool requires Smalltalk installation, which currently is not available for direct download. I needed to order it. Lately I've received an installation CD directly from the vendor, Cincom. The installation is straightforward but it took me some effort to figure it out how to run the tool. The download page says: "On all platforms you can get Softwarenaut working by first installing Cincom Smalltalk and then opening with it the following images: Softwarenaut-3.146.zip (Feb 3, 2011)". I tried to load the image file from the Visual Works GUI with no success. Only when I've associated (in Windows) the image file extension

A: We apologize for not having made it easier in the first place to test the application on Windows. We have provided an executable for the Windows platform. All you have to do is download the archive that can be found at: http://scg.unibe.ch/softwarenaut/downloads . You can now run the tool without the need for downloading and installing Cincom Smalltalk.

Plus every link under Parsed Systems title is broken. Clearly, the website leaves some room for improvement.

A: Again, we apologize for the website not providing a better experience. The links to the demo systems have been fixed and they are available at the bottom of the page of: http://scg.unibe.ch/softwarenaut/downloads.

It looks like the tool does what it claims to do. The GUI and controls are modern and intuitively clear. The tool deserves much credit in comparison with, e.g. Rigi the authors refer to as an inspiring prototype. Especially I liked filtering and focusing capabilities. A few things about GUI:

The placement of the "Softwarenaut? (get the latest/specific version) menu items is questionable: the user usually expects at the first position the "File" menu item (actually it's "Models" in our case). Those ones should go to the "Help" menu item.
The item "Clear all caches? is also looks out of place.

A: This were good observations. We have modified the menus to take them into account.

- The documentation on the tool's website is very succinct if not present at all.
- The biggest minus point of the tool is almost complete absence of the help system: nothing is present except a tiny "Quick start", which is quite cryptic in its "Usage" section. After reading it I still don't understand how to load a new model. It's not clear

what "Moose browser" is and where to find it.
- The absence of a help system is somehow mitigated by quite intuitive context menus and views.

A: We thank the reviewer for pushing us to review our help system. The new system is available online at: http://scg.unibe.ch/softwarenaut/book and addresses topics from installing the tool, to loading a new model, and analyzing a model.

Meanwhile a few examples are preinstalled and readily available in the corresponding window, including the Softwarenaut system itself.
Also, playing with the tool I encountered a couple of times Smalltalk exceptions popping up (without the tool crush though).
To conclude, as the authors admit it in the "Quick start" window, it's still a prototype but prototype promising to become a very useful architecture recovery tool.

A: We thank the reviewer for her patience in working through the problems of the tool. We hope that the changes we have made did address most of the concerns of the reviewer.

Reviewer #2: The paper presents Softwarenaut, an interactive tool for architecture visualization. The tool provides mechanisms for exploration and filtering, which help the user to perform architecture recovery and quality assessment.

Softwarenaut is impressive, and now that Cincom Smalltalk is readily available, the tool is accessible to researchers and practitioners. I had some troubles with the Mac application, but nothing to prevent me from recommending that others try the tool. My biggest issue with the tool is with its supporting web page. The authors do not provide much (textual) detail. For example, I had trouble with the Sourcerer DB integration, but the web page provided no help. Perhaps the authors can prepare an FAQ-like list of potential issues and solutions?

A: We have prepared a better documentation on the new website of the tool. Moreover, the individual pages of the documentation allow for users to post comments and questions so the documentation can be improved live.

The paper is basically acceptable, but I think that the authors should make improvements in key areas. My primary suggestion is that the paper should not weight everything equally. That is, if the evolutionary analysis is particularly novel and interesting (and I do think that it is particularly interesting), you should emphasize aspects of the tool that deal with evolution.

A: We appreciated this observation. We have created a separate section of the paper which talks about evolution.

Issues/comments:

- Figure 2 and its description do not match.
- "A data structure that allows for this." --- what is "this"? The ability to represent explicit and implicit relationships? Or perhaps the ability to aggregate horizontal relationships? Please clarify.
- What exactly is a Higraph? That is, what makes it a "Higraph" (rather than a "graph")? Is DMPHigraph a different data structure, or just a specific instance of a Higraph? Please mention sooner that Figure 3 is a diagram of a DMPHigraph.

* Regarding Figure 3, why is "Class" a Composite Entity but not a leaf entity? I thought it could be either.
* Also, two of the three explicit relationships involve classes, but you state that "explicit relationships usually exist only between the leaf entities". Isn't this a contradiction?

- With regard to Explicit Relationships, how to you handle invocation of polymorphic methods? For example, suppose I have two Java classes A and B, which extends A. Method foo is defined in A and overridden in B. In method bar of class C, I call foo via an A reference. I assume the explicit relationship is between C.bar and A.foo. Is the (potential) relationship between C.bar and B.foo explicit, implicit, or not handled by the tool?

A: We also rewrote the section on the hierarchical graph usage in Softwarenaut. We have reworked the diagram and the associated discussion. In the process we have addressed all the previous observations.

With regard to Composite Entities, what might we cluster? Other Composite Entities, or any kind of Entity?

A: It’s specified now: we assume the clustering of the leaf entities.

Figure 4 omits the leftmost panel, which is a bit disconcerting. I had the tool open as I read the paper a final time, and I wondered (until seeing Figure 10) how far behind the tool the paper is. So, maybe include the full set of panels in Figure 4.

A: The new screenshot includes the left panel.

The formalisms in 4.1 are more confusing than helpful. Perhaps stating the notation and also the meaning of each symbol for each operation would help.

A: We obliterated the formalisms.

* Badly worded sentence: "Only when one has understood.the story of the view."

A: Rephrased to: “One can understand the message only when he has understood all the nouns and the verbs.”

* To me, the evolutionary aspects of the tool are most interesting. Indeed, the authors note toward the end of the paper that functionalities related to evolution are among the most novel and most promising aspects of the tool. So, I am disappointed that Evolution Filmstrip is given only one paragraph. I have many questions about this feature, but I could not even find it in the tool. Again, textual tutorials on the web page would be helpful, because I do not want to watch the entire video(s) just to find out how to pull up a window.

To address these observations we have done the following:

A: We provided an individual section entitled “Evolutionary Analysis” in which we expand the discussion on evolutionary analysis and its use in architecture recovery.

A: We have enriched the documentation with information on how to load multiple models in softwarenaut for multi-version analysis (http://scg.unibe.ch/softwarenaut/book/importing-a-system/importing-for-multi-version-analysis), and we provided a section in the documentation about the evolution filmstrip (http://scg.unibe.ch/

softwarenaut/book/interactive-exploration/detail-panel/the-evolution-filmstrip)

Anyhow, Figure 7 leaves me wondering how much I can actually do with the evolution filmstrip. What if a relationship disappears and then reappears later. Can the tool detect this, or would it be unaware that the two relationships correspond?

A: Yes. In our article in which we introduced the filmstrip ("Exploring Inter-Module Relationships in Evolving Software Systems") we talk about this.

* Again, if Evolutionary filters are particularly novel, emphasize them (and expand the discussion accordingly).

A: Thank you for the idea. We have added the discussion on evolutionary filters in the Evolution Analysis section.

* Perhaps you can add a couple of sentences explaining why other metamodels (e.g., Datrix, Dagstuhl, the language-independent part of Columbus) and exchange formats (e.g., GXL) were not considered.

A: The reason for using FAMIX was to be able to integrate in our immediate research environment.

* Section 6 is the weakest in the paper. See the next section of comments for a bit more detail.
* Section 7 mentions that the primary difference between Creole and Softwarenaut is the ability to perform evolutionary analysis. Again, please provide more discussion of evolution in the paper.

A: We have provided an individual section on evolution.
We have rewritten section 6.

* Also in Section 7, the distinction between snapshot-based and commit-based models is unclear to me. Isn't a commit just a snapshot? Similarly, can't a snapshot be a commit?

A: We rewrote that part to clarify. What we meant was that we build a complete model of several versions of the system while they have an evolutionary model that seems to compute only the differences.

* The discussion of Jazz vs. Softwarenaut is a bit weak. If the main difference between the tools is that they have different project goals, maybe you should focus more on how they are similar.

A: We rewrote this part.

* One additional comment that I have deals with the relationship between the architecture evolution and clone evolution. The problems seem closely related, and it seems to me that you can learn from the challenges/solutions identified by the clone evolution community as you continue with your work.

A: Thank you for the advice. In the future we will look into clone evolution research for ideas.

Section 6 issues/comments:

* Typo: "we provided a framework in which masters projects to be developed."
* Section 6.1 should be rewritten and expanded. Though I do not expect a controlled experiment here, I do expect a bit more rigor and detail. For example, do you have any background information about the participants (e.g., how many had experience with similar tools, or with modeling tools in general?) or do you have more thoughts on the results of the informal study (e.g., what is your opinion regarding questions 2 and 3?)?

A: We have rewritten section 6.1 to provide a more rigorous treatment of the usability study. It's now section 7.2.

* Section 6.2 is too vague to be of interest. What projects did you depend on? Further, you provide no insight regarding why/how your experience is unique or unexpected.

A: We have expanded on this section.

* Section 6.3 does not need to be expanded, but the materials to which it refers (i.e., documentation on the web page) do need to be.

A: Done. We would like to thank again the reviewer for his valuable feedback.

Reviewer #3:

==== Summary ====

The paper presents Softwarenaut, a reverse engineering tool that supports exploration of static structure of a software system that helps in understanding its underlying architecture. The contents of the paper can be summarized as follows:
- some details on the underlying model (DPMHigraph) and operations (navigation, creating and sharing architectural views of software under study) of Softwarenaut.
- demonstration of the supported operations using screenshots of Softwarenaut in action explain various features available in it
- an introduction to the building blocks of Softwarenaut that were already existing (fact extraction tools, famix, moose etc
- a short section on experience related to building Softwarenaut itself and, some collaborative work that came out of Softwarenaut research (SPO, semantic clustering)

==== Comments ====

* Relevance: Is this paper relevant with respect to the system?

[COMMENT] Yes, the paper is relevant with respect to the system.

* Motivation: Why is this tool developed?

[COMMENT] The motivation seems to be clear. Softwarenaut was developed to help understand and keep track of the underlying architecture of large evolving software - a very common problem in software engineering.

* Application: Are the authors explicit about the application area of the developed tool?

[COMMENT] Yes, the authors explicitly mention the tool is for architectural recovery, that could be considered an area related to software understanding and maintenance.

* Objectives: Are the objectives of the system clearly stated?

[COMMENT] The objective seems to be clear. Softwarenaut is presented as a tool that (supposedly) helps in tackling two problems related to software architecture: (1) recovery - how do i understand or discover the underlying architecture as my system grows, and (2) assessment - how do i know if the architecture is good.

* Tool builders issues: short discussion of issues encountered during tool building that may be relevant to other tool builders.

[COMMENT] There are shore discussions that address this (Sections 5/6), but I think that can be improved (please see my comments [1]further below).

* English: Is the paper written clearly with correct English?

[COMMENT] Yes.

Regarding the tool itself we ask the reviewers to try to install and use the tool and take a look at the installation and user documentation, implementation, tool web-site, tutorials, examples or whatever other support the authors would have prepared for the reviewers and readers. In particular, we ask the reviewers to pay attention to the following quality aspects for the tool:

* Installation documentation, ease of installation.

[COMMENT] Installation was straightforward. Download, install, and run. It seemed that the installation required Cincom's Visualwork to be installed on the system as I got some exceptions complaining about the home path of its installation. I got the Cincom CD after a month I applied for it, however even after installing it, I did not see much difference in my usage experience. My platform was Mac.

* Quality of user documentation.

[COMMENT] Beyond some web pages, and a demonstration video there was no user documentation. I saw features in the paper that I had no idea how to try in the version of the Softwarenaught I downloaded (evolution strips, omniprescent module filter). I would say this tool was lacking in a good quality user documentation. It'd be very helpful if there was a separate document or perhaps an online demo (video) for each of the tasks you could achieve with Softwarenaught as presented in the paper.

A: We have greatly improved the online documentation which you now can find at: http://scg.unibe.ch/softwarenaut/book

* Quality of the examples provided by the authors/system developers These examples should be relevant to the application area.

[COMMENT] The additional examples that were available and usable were the preexported mse files. Those files were missing from the download area, the author was responsive to fix the link. However, once I downloaded those files, first with the current version of Softwarenaught from the web, I constantly ran into numerous exceptions while doing different operations. I have attached a separate document with screenshots of those exceptions.

[COMMENT] While it is easy to load up Softarenaught with sample files and see the available views (at least for the Softwarenaught's model itself for example), there was not much I could do beyond that. For example, I wanted to explore Lucene's architecture using the provided sample mse file. I have used Lucene for various projects and was really interested in understanding its architecture focusing on certain features in it (for example index merging). Once I expanded the single box that was presented to me, the view the exploded with zillion of nodes (it seemed like 'explode' not 'expand'). I had no clue how to filter them, and get really nice views as shown in the paper, as selecting some existing filters did not help me reduce the clutter. Zooming, layout, filtering all ran into exceptions. After couple of attempts I gave up.

A: That was a bug that we fixed. We hope that you can now analyze Lucene without troubles.

I was really excited to see SourcererDB integration too. However, I was really surprised to see just couple of projects listed (unlike 10k+ as the website of SourcereDB mentions). I had some success in browsing some models from SourcererDB, perhaps they were not as complex as Lucene.

A: Unfortunately we depend on the developers of SourcererDB and they are still rebuilding their database. Not much we can do there.

I also think the performance of the tool is beyond acceptable. I had to wait (perhaps few more seconds than I expect) with no indication of progress for operations like filter. The system just seemed to be unresponsive for a bit, then when it'd be responsive I again saw no effect of the filters I applied.

A: Filters work for us. We would need more details on how to replicate the error.

Bottomline: There were far too many exceptions than I expect, and not enough guides to walk me through a process that would let me recover an architecture of the system I am interested in, even when I was already familiar with it to some extent (e.g. lucene).

A: We fixed the bug that was happening when analyzing Lucene. Now you should be able to analyze it!

* Applicability to the domain.

[COMMENT] I think the ideas in the paper are interesting and probably useful for architectural recovery, but Softwarenaught has to prove itself to be a usable tool for the objectives that are stated. At least as a new user I had hard time doing anything significant with it except opening intendedsamples and looking at predefined views. I believe there is not enough evidence that Softwarenaught can really be an ideal tool for architecture recovery that real practitioners can use during maintainence and comprehension tasks. Next step could be more formal experiments and (longitudinal) user studies.

==== RECOMMENDATON ====

Based on the above aspects the reviewers will make an overall judgement: "publish as is", "minor revisions", "major revisions", or "reject".

I would recommend a "Minor Revision" for this submission, where I suggest the authors to address the following issues:

[1] I liked the discussion on Architectural Considerations and introduction to Tool-Building Consideration. In the spirit of this special issue of SCP, I think it would be better if those sections could be expanded a bit. Especially these issues are worth being elaborated:

-- Summarize the research papers a bit [24,33,15,14], with some insight on how research progressed across work related to Softwarenaught. What lessons were learnt, what contributions were made ?

A: Done.

-- More details on collaborative work with Softwarenaught. What made such collaboration possible? Did it require some specific work/redesign ? Were there extension options APIs/plugins for Softwarenaught that enabled building complementing tools? Was it because of the Famix model or some other common framework ? How can someone else interesting in contributing extend Softwarenaught ?
-- It might be worth rearranging discussion in terms of upper layers (framework Softwarenaught uses) such as Famiz, fact extrators; then the design of Softwarenaught; and, complementing/collaborative tools such as clustering work, and new work with SourcererDB (as mentioned in the website)

A: We tried to provide more details here.

[2] Clearly state the limitations with usability and whether Softwarenaught achieves its goal. There is really not much work done on evaluation of the tool itself with users of various expertise level, and with real-world large software systems. How'd we know an approach such as Softwarenaught is good for practitioners? It is questionable that the two goals that Softwarenaught was set out to achieve have really been achieved. This seems to be a serious limitation, and the authors should address/acknowledge that in some way in the revision. **Section 6.1 seems to be an early (exploratory study) step towards this, however the study seems to be quite informal and not in-depth (at least the explanation is not).**

A: We expanded section 6.1. We have provided information about the limitations of the tool in section 6.1. We have listed limitations that participants in our study expressd, as well as limitations that we consider that need to be addressed (the node explosion).

I also have some clarification questions that I believe should be answered in the revision:

[3] One important question I kept asking while reading the paper was, what is the research contribution here? Yes, it does seem like a useful tool based on some known existing idea and body of research. But, if I have to take Softwarenaught and ask what is the underlying research hypothesis and has it been validated, I was left unanswered.

A: We wrote the article focusing on presenting the tool as a whole rather than presenting a “scientific study” with a research question and validation. As we have said, the tool is only the vehicle for past and future research.

[4] Being based on the Famix metamodel, it seems that Softwarenaught assumes that a software project is just a set of classes (or other OO entities) and relations. And, it seems that the model is a result of some sort of static analysis (fact extraction). So, will it be a problem when all relations cannot be inferred statically ? What about projects where relations span across multiple artifacts: xmls, configurations, properties, DSLs, etc ?

A: As long as the fact extractors can extract entities, relationships, and provide a hierarchy between them, Softwarenaut can be used to visualize them. It is however true indeed that until now we have focused mostly on

[5] Page 4, Lines 39 - 45 mention mc2, mc5, c1 etc. These are not found in Fig 2. Are they supposed to be CA1, CA2 .. etc instead ?

A: We have fixed this.

[6] Page 8, Line 55. How do you filter omnipresent modules (better to have a short description on what omnipresent moldules are)

A: We have to add back the filter for omnipresent modules. It is a module filter.

[7] Figure 6 mentions one can navigate to the source code of the system, is that only for those written in smalltalk ? If I only have the mse file, I assume I won't be able to nagivate the source code.

A: That is an astute observation: in Smalltalk you can always navigate to the source code. For systems written in other languages, if you have the sources on your disk and your fact extractor annotates entities with the path to the source you can indeed navigate to the source also from Softwarenaut. There is more work to be done on this in the future like downloading the source files on demand...

[8] Page 10 mentions 'evolution filmstrip'. I think I was not able to see that view in Softwarenaught. How do I get to that view ?

A: We have put forth instructions here: http://scg.unibe.ch/softwarenaut/book/interactive-exploration/detail-panel/the-evolution-filmstrip

[9] Page 11, Line 33. What are 'weak dependencies' and 'small modules' ?

Improvements in supporting materials

[10] I appreciate that the authors have a dedicated web-site for the tool that seem to be updated when needed. I would suggest the authors to produce few more details usage guide targeting new users that shows them how to use Softwarenaught to achieve certain tasks on bigger examples (eg Lucene -- ie show me how to get to a nice view from exploded graph with zillion nodes/edges). Since, I ran into multiple exceptions, I wonder if there is any additional setup that needs to be done wrt Cincom smalltalk, if yes please document those too.

A: We have updated the website. In the same time, the Lucene incident was a problem that we hope will not happen again.

We would like to thank the reviewer for the detailed feedback.