What Ails End-User Composition: A Cross-Domain Qualitative Study

Vishal Dwivedi(✉), James D. Herbsleb, and David Garlan

School of Computer Science, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, USA
{vdwivedi,jdh,garlan}@cs.cmu.edu

Abstract. Across many domains, end-users need to compose computational elements into novel configurations to perform their day-to-day tasks. End-user composition is a common programming activity performed by such end-users to accomplish this composition task. While there have been many studies on end-user programming, we still need a better understanding of activities involved in end-user composition and environments to support them. In this paper we report a qualitative study of four popular composition environments belonging to diverse application domains, including: Taverna workflow environment for life sciences, Loni Pipeline for brain imaging, SimMan3G for medical simulations and Kepler for scientific simulations. We interview end-users of these environments to explore their experiences while performing common compositions tasks. We use "Content Analysis" technique to analyze these interviews to explore what are the barriers to end-user composition in these domains. Furthermore, our findings show that there are some unique differences in the requirements of naive end-users vs. expert programmers. We believe that not only are these findings useful to improve the quality of end-user composition environments, but they can also help towards development of better end-user composition frameworks.

1 Introduction

Increasingly, end-users rely on computations to support their professional activities. Although in some cases turnkey applications and services are sufficient to carry out computational tasks, there are many situations where users must adapt computing to their specific needs. These adaptations can take many forms: from setting preferences in applications, to "programming" spreadsheets, to creating orchestrations of services in support of some business process. This situation has given rise to an interest in end-user programming [13], and, more generally, end-user software engineering [7] or end-user computing [6]. This emerging field attempts to find ways to better support users who, unlike professional programmers, do not have deep technical knowledge, but must somehow find ways to harness the power of computation to support their tasks.

One important subclass of end-user computation arises in domains where end-users must compose existing computational elements into novel configurations. In these domains end-users typically have access to a large number of

S. Barbosa et al. (Eds.): IS-EUD 2017, LNCS 10303, pp. 66–83, 2017.
DOI: 10.1007/978-3-319-58735-6_5

Table 1. Example composition environments across different domains

Type	Compositions
Astronomy	Electromagnetic image processing tasks [1]
Bioinformatics	Biological data-analysis services [9]
Digital music production	Audio sequencing and editing [10]
Environmental Science	Spatio-temporal experiments [17]
Geospatial Analysis	Interactive visualization of geographical data [12]
Home Automation	Home devices and services [8]
Neuroscience	Brain-image processing libraries [2]
Scientific computing	Transformational workflows [16]
Socio-technical Analysis	Dynamic network creation, analysis, reporting and simulation [15]

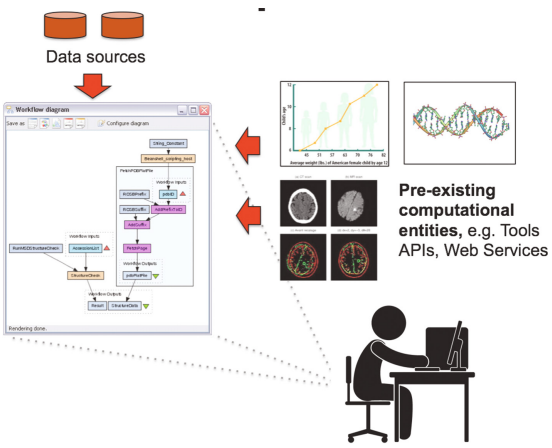


Fig. 1. Compositions using Taverna environment.

existing applications and data sets, which must be composed in novel ways to perform various domain-specific tasks besides generate reports and miscellaneous research findings. Table 1 lists examples of some of these domains and the types of compositions end-users build.

Innovative research in these domains often requires scientists to compose a large number of tools and apply them to data sets to perform experiments and diagnose problems. Figure 1 illustrates Taverna - a popular composition environment that is used to create compositions by combining existing web-services discovered through various service registries.

Unfortunately, assembling such elements into coherent compositions is a non-trivial matter. In many cases users must have detailed low-level knowledge of things like application parameter settings, application invocation idiosyncrasies, ordering restrictions, and scripting languages. Further, it may be difficult for end-users to determine whether a set of components can be composed at all,

and, if not, what to do about it. For example, differences in data encodings may make direct component composition infeasible without the inclusion of one or more format converters. Even when a legal composition can be achieved, it may not have the performance (or other quality attributes) critical to the needs of the end-users. Across many domains, such problems with end-user composition has led to a large number of a large number of composition environments, out of which only few are successfully adopted by these communities.

The above factors have spurred a number of research projects aimed at understanding of end-user programming and improving the usability of the programming environments. Andrew Ko et al [7] surveyed the software engineering challenges faced by end-users, including a framework for handling requirements, as well as making decisions about design, reuse, integration, testing, and debugging for end-user software engineering. Judith Segal [16] in his work has studied "professional end-user developers" — people such as research scientists who work in highly technical, knowledge-rich domains and who develop software in order to further their professional goals. In his studies of various professionals, Segal discovered that the key challenges for such end-users was not learning the programming languages but creating and sharing knowledge and various cultural aspects of the e-sciences ecosystem. And more recently, there have been a number of research efforts to understand the ecosystems [10] and problems related to reuse and sharing of workflows [4].

Much of the research effort towards helping professional end-user developers has focused on the software development processes and user studies to understand sharing and reuse across environments like Loni Pipeline [4] and projects such as Workflow4Ever[1] to understand how end-users can reproduce their workflows. However, we still need a better understanding of end-user composition as a programming activity, the problems end-users face in performing that, and the challenges in developing quality end-user composition environments.

In previous research, we have developed a technique called "end user architecting" and a software framework to support end-user composition [5]. As a part of this work, we formed some initial hypotheses about key critical barriers to end-user composition — something that we identified through exploratory studies and system implementation in three domains: dynamic network analysis, brain imaging, and geospatial analysis. However, for much of this work, we were the developers of these environments. A confirmation of these hypotheses required a principled study of end-users, perhaps of different composition environments.

To do this, we designed a qualitative study where we interviewed users of some carefully selected composition environments across very different domains. We asked the end-users about their experiences in creating compositions, the challenges they faced and the characteristics of composition environments that support or inhibit composition tasks.

[1] https://www.force11.org/node/4708.

Specifically, we have had the following four research questions:

Q1. *For what purposes do end-users use composition environments?*
We wanted to see for what activities did end-users used their composition environments. Was it merely drawing and execution or were there more types of tasks involved?

Q2. *What composition tasks do end-users perform and what difficulties they face in those tasks?*
Through open questions based on a sample composition the end-users drew, we wanted to understand the types of composition activities and the general difficulties end-users faced in creating compositions.

Q3. *What key quality features do end-users want in their composition environments, and what is the relative importance of these features?*
We had some assumptions about the quality problems and we wanted to test if these were indeed true.

Q4. *Does "skill level determine relative importance of the environment features and the quality problems?*
We wanted to determine if expertise with composition environments played any role in the kinds of problems end-users faced. To quantify skill-level, we assumed experts were people who had more than one year experience using their composition environments while anyone with less than 1 year experience was considered a beginner.

This paper presents the results from our study focusing specifically on these questions. In Sect. 2, we describe our initial hypothesis about barriers to end-user composition. In Sect. 3, we describe the research design of our study. In Sect. 4, we present the key five findings of our study. Finally, we have a discussion about the possible implications of these findings and some recommendations for composition environment developers to build quality end-user composition environments.

2 Barriers to End-User Composition

As noted above, a large number of domains depend on composing computational elements to accomplish some domain specific tasks. A number of research and practitioner-based efforts have produced platforms that provide end-user tools for composition, reuse and execution within these domains. Furthermore, there exists a large number of component repositories and environments that support computational models, such as workflow execution, widget composition, data exploration or music synthesis and composition.

While many of these platforms have been successful, there are others that have failed to make a mark. In our previous work [5] we hypothesized that the following quality barriers impact the adoption of end-user composition environments:

1. **Excessive technical detail:** Creating compositions currently often requires knowledge of myriad low-level technical details, such as data formats, parameter settings, file locations, ordering constraints, execution conventions, scripting languages, etc. As Fig. 2 illustrates, brain imaging research using FSL

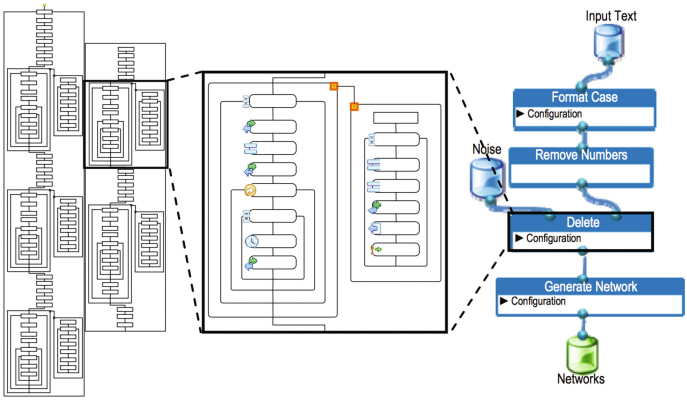


Fig. 2. An example end-user composition mapped to a BPEL language script

toolsuite[2] requires a user to understand and create detailed execution scripts that specify how to configure each of the constituent tools, which may have dozens of configuration parameters. As another example, in the domain of intelligence analysis a typical composition that involves two logical steps, but is executed in the context of a service-oriented architecture (SOA), requires the end-user to specify a Business Processing Event Language (BPEL) script shown in Fig. 2 [15]. The script requires the user to explicitly specify low-level details that handle control flow, variable assignment, exception handling, and other programming constructs.

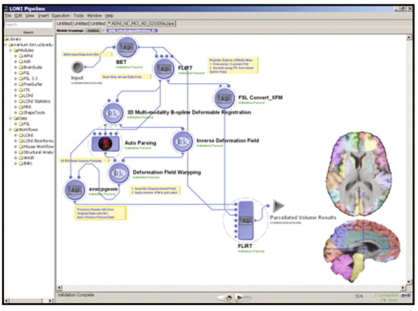
2. **Inappropriate computational models:** The computational models provided by typical execution platforms, such as SOA, may require end-users to map their tasks into a computational vocabulary that is quite different from the natural way of decomposing the task in that domain. For example, tasks that are logically represented in the end-user's mind as a workflow may have to be translated into the very-different vocabulary of service invocations executing on a SOA, as illustrated in Fig. 2.
3. **Inability to analyze compositions:** There may be many restrictions on legal ways to combine elements, dictated by things like format compatibility, domain-specific processing requirements, ordering constraints, and access rights to data and applications. Currently, discovering whether a composition satisfies these restrictions is largely a matter of trial and error, since there are few tools to automate such checks. Moreover, even when a composition does satisfy the composition constraints, its extra-functional properties — or quality attributes — may be uncertain.
4. **Lack of support for reuse:** An important requirement in many communities is the ability for professionals to share their compositions with others in those communities. For instance, brain researchers may want to replicate the analyses of others, or to adapt an existing analysis to a different setting (e.g.,

[2] www.fmrib.ox.ac.uk/fsl.

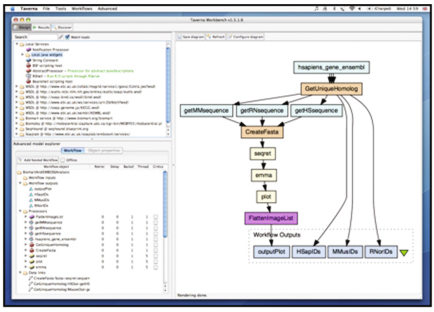
executed on different data sets). Packaging such compositions in a reusable and adaptable form is difficult, given the low-level nature of their encodings, and the brittleness of the specifications.

5. **Impoverished support for execution.** The execution environment for compositions is often impoverished. Compared to the capabilities of modern programming environments, end-users have relatively few tools for things like compilation into efficient deployments, interactive testing and debugging (e.g., setting breakpoints, monitoring intermediate results, etc.), history tracking, and graceful handling of run-time errors. This follows in part from the fact that in many cases compositions are executed in a distributed environment using middleware that is not geared towards interactive use and exploration by technically naive users.

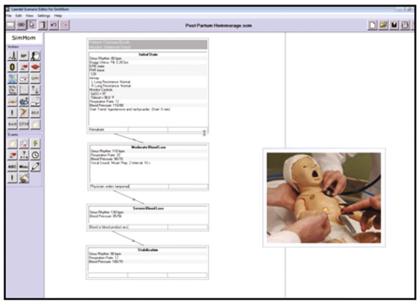
These quality barriers were identified based on our exploratory studies in three domains: dynamic network analysis, brain imaging, and geospatial analysis [5]. In theory, we believed that these would hold for other domains too. Our multiple-case study is designed to evaluate if this is indeed true. We were also interested in discovering whether any other important quality dimensions or important observations would help in the design of successful end-user composition environments.



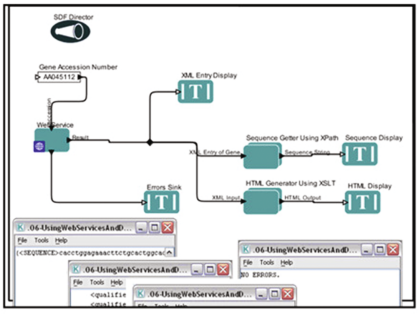
Loni Pipeline (Data flow)



Taverna (Data flow)



SimMan3G Scenario Editor (State models)



Kepler composition environment (Mix of Data flow & Control Flow)

Fig. 3. Composition environments under study.

In the next section, we describe our qualitative study of the four composition environments (Fig. 3).

3 A Qualitative Study to Investigate Problems Faced by End-Users in Designing Compositions

We chose an exploratory, qualitative research method that aims to understand how end-users used their composition environments across different domains and problems faced by them. Our method consists of three main phases:

- The case selection and protocol design phase, in which we developed the research protocol and identified a diverse set of composition environments with different composition styles and application domains.
- The interview phase, wherein we elicited responses from the selected end-users.
- The qualitative data analysis phase, in which we coded the interview transcripts and systematically drew inferences from the data.

Next, we describe the 3 phases of our study.

3.1 Case Selection

As shown in Table 1, composition environments today use a wide variety of composition models, varying from dataflows (e.g., Loni Pipeline and Taverna) to publish-subscribe (e.g., Ozone Widgets) to state-based transitions (e.g., SimMan3G simulation) to mix of composition styles (e.g., Kepler). An important consideration for our study was to explore the differences across these domains and composition models. For instance, did end-users face the same problems while designing workflows as they did while composing states? We selected 4 candidate environments that were quite different in their domain of application and composition models. Besides this, we conducted a pilot study using an industrial composition environment called "Appian modeler", which is a dataflow based composition environment.

We provide a brief description of these composition environments below:

1. **Loni Pipeline:** is a dataflow-based composition environment for neuroscience workflows. The compositions in Loni Pipeline environment reference data, services and tools as components that can be assembled together through a drag and drop interface. As per a software usage survey[3] conducted by NeuroDebian in 2011, Loni Pipeline was one of the top 20 environments in the neuroscience domain.
2. **Taverna:** is a dataflow-based composition environment for designing and executing web-services compositions. Initially designed for bio-informatics, Taverna is currently being used by users in many domains, such as bioinformatics, cheminformatics, medicine, astronomy, social science, music, and digital preservation.

[3] http://neuro.debian.net/survey/2011/results.html.

Table 2. Study participants.

Tool	Participant	Expertise level
Appian modeler	P0 (Pilot)	Beginner
Taverna	P1	Beginner
Taverna	P3	Expert
Taverna	P4	Expert
SimMan3G	P5	Beginner
SimMan3G	P6	Expert
SimMan3G	P7	Beginner
Kepler	P8	Beginner
Kepler	P9	Expert
Loni Pipeline	P10	Beginner
Loni Pipeline	P11	Expert

3. **SimMan3G:** is a state-based patient simulation system that facilitates health-care training by simulating real-life medical scenarios such as a cardiac arrest, breathing complications and change of vital signs on the high-fidelity manikins. Medical training professionals can combine a sequence of such activities to create a medical scenario (such as an asthma attack) and the complications that go along with it. These activities can be currently programmed in a composition and automatically executed on a manikin or a simulator.
4. **Kepler:** Kepler is a composition environment for designing and executing scientific workflows that uses a mix of dataflow and control flow semantics. Using Kepler's graphical user interface, users can compose various analytic components and data sources to create a scientific workflow. The Kepler software helps users share and reuse data, workflows, and components developed by the scientific community to address common needs.

For the composition environments described above, we recruited 10 participants (plus one additional for the pilot) who had a different degree of expertise in using the composition environment. The average total interview time per participant for each interview was about 35 min. Our participants consist of a mix of beginners (with less than a year experience) and experts (who had been using their composition environment for many years). Table 2 shows the list of participants for the study. It is to be noted that our "expertise level" criteria was fairly subjective and was reinforced during the interview through direct questions about the participants background and the level of their experience and expertise using their composition environments.

3.2 Semi-structured Interviews

For our qualitative study, we followed a semi-structured interviewing discipline [3], which means that although the interviews were guided by an explicit interview protocol that defined the general topics that the interviews would examine, we were free to devise new questions to further probe interviewees on specific subjects.

All subjects were asked to either draw a composition (as a homework task), or reproduce an existing composition they had previously drawn. During the interview, all participants were asked to open up their composition and they were interviewed about their experience writing that composition. The general technique used was to start with open-ended questions such as "What problems did you face in creating this composition?", and then ask detailed questions about specific types of problems.

Our interviews consisted of an introductory script to secure informed consent followed by a series of topics to be covered including the following:

- Questions about a participant's role and background and expertise
- Questions about a recently drawn composition (before the interview) that participants needed to open up and use as a recall mechanism
- Questions about features used to create that composition
- Questions about problems faced and quality issues of the environment
- Ratings of quality issues
- Suggestions: how can limitations be addressed?

We instructed participants to speak out loud and explain their actions while working with the composition environments. The recorded audio statements of participants were further transcribed and analyzed.

3.3 Data Analysis and Interpretation

Given the exploratory nature of our research questions, "Content Analysis" [11] is the main analytic method used in our study. The content analysis technique allows building an understanding of underlying reasons and motivations of participants while using unstructured or semi-structured data (such as interviews).

We recorded all participant interviews and used Amazon Turks to transcribe the audio into text, which needed some post processing. We used coding theory [14] to link the findings about end-user preferences to the interview dataset and validate whether our observations were consistent. In particular, we employed a two-cycle coding method: in the first cycle, we applied the "hypothesis coding method to our dataset using the predefined code list. In the second cycle, we applied axial/pattern coding to discover patterns from the dataset [14].

A selection of sample 1st cycle codes is listed in Table 3. As a second-cycle coding activity, we identified patterns and selective heuristics that led to some of the key findings for the study that we discuss in the Findings section.

Table 3. Sample (first-level) codes for the study.

1st cycle codes	
1. Composition motives (mot) • Simulation (mot:simulation) • Experimentation (mot:experiment) • Teaching (mot:teaching) • Automation (mot:automation) • Other (mot:other)	4. Resolution of problems (res) • Analysis tools (res: tools) • Intuition (res: intuition) • Execution (res: execution) • Reference Documentation (res: docs) • Other (res: other)
2. Nature of Composition (nat) • Computation model (nat: compModel) • Abstraction level (nat: abstractionLevel) • Other (nat: other)	5. Desired Feature (des) • General Purpose (des: general) • Tool-specific feature (des: specific)
3. Quality issues with composition environments (issue) • Technical detail (issue: techDetail) • Reuse support (issue: reuseSup) • Execution support (issue: execSup) • Analysis support (issue: analysis) • Computation model mismatch (issue: compMismatch) • Other (issue: oth)	6. Skill level of end user (skill) • Beginner (skill: beginner) • Expert (skill: expert) • Unknown (skill: Unknown) 7. Rating (rating) • Highly important (rating: highImp) • Low importance (rating: lowImp) • Unknown (rating: Unknown)
Other codes...	

4 Findings

In this section we revisit the research questions we identified in Sect. 1 and how content analysis helped us to find answers to those questions. As an outcome of our analysis, we discuss 5 key findings that provide some insight into how end-users use composition environments and what problems they face. We believe, these could be a basis for further research and improvements to composition environments.

4.1 Finding 1: Goals of End-User Composition — End-users Use Composition Environments Not only to Perform composition Tasks, but They Also Serve as Experimentation And learning Tools

To address *ResearchQuestion1* ("For what purposes do end-users use composition environments?"), we evaluated the first-cycle attribute codes, where participant responses point to a number of "Composition Motives". While the frequency of codes varied across environments, the general finding was that participants used the environments as learning and experimentation tools. Table 4 lists the breakdown of general composition motives for the participants.

While the frequency of occurrences of codes was not the most interesting observation, what was more relevant was "how" end-users performed the

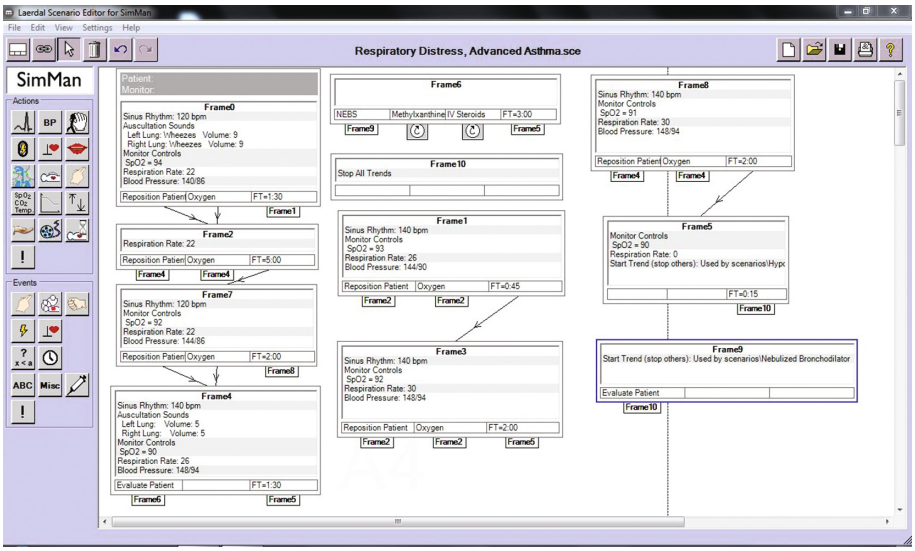


Fig. 4. A respiratory distress scenario using SimMan3G (Participant P6)

Table 4. General layout of composition motives.

	Frequency (Coding Units)	Frequency (Interviews)	Frequency (Participants)	Examples
mot: simulation	12	10	7	Medical Simulation, Workflow simulations, etc.
mot: experiment	22	10	10	Tutorials, Adapt compositions, etc.
mot: teaching	4	10	1	Medical Simulation teaching
mot: automation	14	10	10	Workflow automation
mot: other	32	10	4	Exploration, Debug, Reuse, Learning, etc

composition tasks and the qualitative reasoning in doing them. As an example, Fig. 4 shows a composition on SimMan3G simulator for an advanced asthma attack scenario simulated on a high-fidelity manikin. The typical problem in such composition scenarios is fine-tuning the properties about the medicine dosage, oxygen levels etc. Much of the composition activity for this composition environment involved going to-and-fro between composition and execution to understand the individual components. While drawing a composition may be important, much of the effort was spent in interactive learning and exploration.

To exemplify how important is this exploration and experimentation activity, here are some statements collected during the interviews:

- "Before we begin simulation, we follow a worksheet to organize our thoughts and we constantly refer to that to find out who the patient is, what kind of position they are in [properties] .... we go to-and-fro between this worksheet, drawing, and execution to learn more..." [Participant P5 about Medical Simulation scenario]
- "The biggest problem is not connecting everything together but to understand what's going on" [Participant P1, Taverna]
- "...Sometimes there is a need to take a different model and use a different controller, but that means manually figuring out all the signals and even look at code to find what's going on..." [Participant P8, Kepler]

For many participants the composition environments were a mechanism to quickly test some components, or explore an existing composition to learn a concept. This aspect is often missed in many composition environments. Composition environments can further support learning and cognition by providing tutorials, integrating repositories with composition environments (like myExperiment in Taverna).

4.2 Finding 2: Types of End-User Tasks — End-User Composition Involves Multiple Phases, Including: (i) Search and Exploration (ii) Reuse (iii) Construction (iv) Analysis (v) Execution, and (vi) Debugging

To address *ResearchQuestion2* ("What composition tasks do end-users perform and what difficulties they face in writing compositions?"), we evaluated the first-cycle attribute codes, where participant responses point to a number of "Composition Tasks" and "Composition Problems". While first cycle codes provided us instances of tasks such as search, reuse, construction, analysis, execution and debugging (not necessarily in that order), what was more interesting was to find pattern codes such as:

[task: search→task: construct→task: execution→tasks: analysis→task: debugging] for each composition environment. The ordering and membership of such patterns varied, but such patterns were common across all the four environments under study.

Analysis of our interview data showed that end-user composition activity is not a monolithic drawing and execution activity. In fact, the composition activity started with search (on online forums, desktops, component repositories) followed by some level of reuse, experimentation and debugging. External reuse (other people's composition) was rare, and used primarily for learning purposes. But even while drawing everything from scratch, end-users flipped through these phases for a better understanding of the compositions.

While this may seem a not-so-novel finding at first, we can't stress the importance of these phases any lesser. We found almost all the participants going

Table 5. End-user ratings for quality features (Low Importance:1 - High Importance:10)

Tool	Appian	Taverna			SimMan3G			Kepler		Loni Pipeline		Average Score
Participant	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	
Skill-level	Beginner	Beginner	Expert	Expert	Beginner	Expert	Beginner	Beginner	Expert	Beginner	Expert	
Technical detail	6	2	2	2	2	2	6	1	2	4	2	2.5
Reuse support	4	8	9	8	5	2	5	8	4	4	5	5.8
Execution support	4	10	10	8	8	10	8	6	8	8	8	8.4
Analysis support	9	10	9	8	9	8	10	6	7	10	7	8.4
Computation model mismatch	8	2	3	2	9	4	8	6.5	10	2	2	4.85

High importance: Analysis Support, Execution Support
Some importance: Reuse Support, Computational model mismatch
Low Importance: Technical detail

through these phases during their composition activity and they struggled when tool-support was missing for any phase. While some environments had better support for all these phases (e.g., Taverna) and others had limited support for some of these phases (e.g., SimMan3G, which had limited search and exploration capability), in which case end-users resorted to external artifacts like using checklists online forums etc. Not only was it important to support all the phases, but it was also important to allow easy switch between them. Providing support for all these phases is an engineering challenge but it greatly helps the composition activity.

4.3 Finding 3: End-Users Face Some Common Problems Across Different Composition Styles and Domains

To address *ResearchQuestion3* ("What key quality features do end-users want in their composition environments, and what is the relative importance of these features?"), we evaluated the first-cycle attribute codes for the code (quality) issue. While in Sect. 2, we had hypothesized about the key critical barriers for end user composition. We found these to be generally true for all composition environments. As shown in Table 5[4], Analysis and Execution Support were the most important features for end-users as they not only helped in debugging but also learning about compositions. Computation model mismatch played an important role for environments that had a computation style that was very different from composition mechanism. An example was SimMan3G editor, where people thought in terms of sequences and dataflows but had to program events, which was not an easy cognitive switch. Support for reuse was also an important requirement to address composition problems. However, the form of reuse varied across environments. "Self-reuse" was the primary form of reuse. "External

[4] Note that ratings used in Table 5 are not absolute. In this qualitative study, their main role was to help the end-users easily express their preferences.

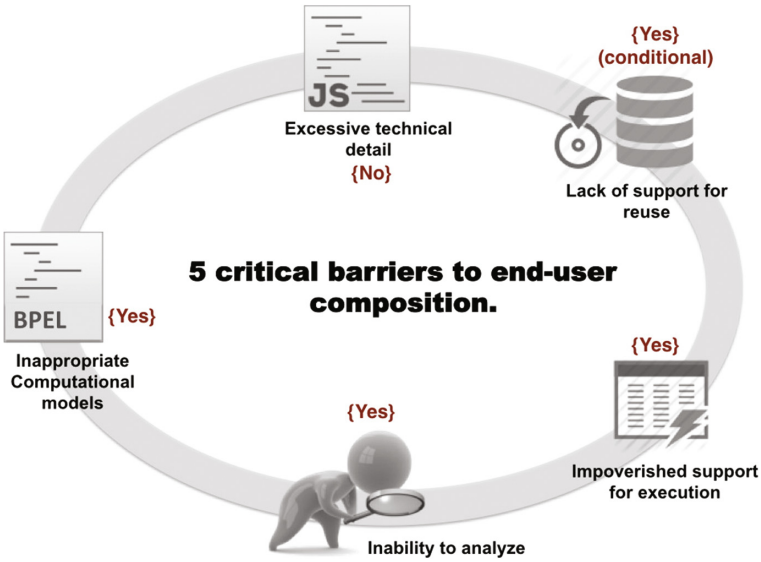


Fig. 5. Our initial hypotheses about end-user composition quality problems.

reuse" via repositories was used mainly used for experimentation and learning. This was not a very surprising result as prior studies have found similar observations [4]. It was slightly surprising that end-users rarely faced any difficulty with the technical vocabulary of the environments. As long as the composition environments provided support for exploration and debugging, the participants were fairly comfortable with the detailed technical vocabulary for all the environments under study (Fig. 5).

4.4 Finding 4: The Skill Level and Purpose Determines the Quality Features Needed by End-Users

To address *ResearchQuestion4* ("Does" skill-level determine relative importance of the environment features and the quality problems?), we looked at patterns of occurrences of [Skills→Quality]. Analysis of our interview data showed some very interesting differences in the preferences of experts and beginners (Summarized in Table 6). We were slightly surprised that beginners were less troubled by the technical details. Perhaps, this had more to do with a visual vocabulary of all the environments. But even in environments like Kepler where some coding knowledge was required, the end-users were comfortable in understanding, if not writing the code. As long as their composition environment allowed them to explore and debug, they did not rate language-constructs and technical details as a barrier to composition tasks. However, in retrospect, this finding is corroborated by similar studies by Judith Segal [16], where he found that programming language complexity was not the key challenge for most professional end-user developers.

Table 6. Preference of Experts vs. Beginners

	Experts	Beginners
Technical detail	Both beginners and experts were comfortable with the level of technical details (Surprising, because we expected beginners to prefer a less-technical vocabulary)	
Reuse support	(1) Prefer self-reuse (2) When there were no repositories there was low preference for reuse	1. Prefer external reuse (for learning) 2. Costly modification implied low preference for reuse
Execution support	(1) Both experts and beginners preferred execution support (2) While experts relied on interactive execution (for debugging), beginners preferred more turnkey execution	
Analysis support	Both experts and beginners prefer more automated checks and custom analysis	
Computational model mismatch	Experts don't find computational model mismatch as issue	Beginners are overwhelmed by a mismatch in computational model

One major difference was the level of reuse for both types of end-users. External reuse was rare, and mainly used for educational purposes. In fact, the most common form of reuse was self-reuse where end-users preferred to use their prior compositions. However, presence of curated repositories changed some of this behavior to some extent. For experts, the bigger concern was not only time spent in learning and exploration, but also trust issues with external compositions.

Here are some statements to describe the nature of reuse being currently practiced:

- "Do you generally reuse your own components or some repository"....."Mostly my own. A vast majority of trends are located in ... folder, I have created over 9 and I have re-purposed them for various scenarios" [Participant P6, an expert SimMan3G end-user describing a self-reuse scenario]
- "I used the myExperiment repository. It helped me a lot to design examples before I could design my own workflows" [Participant P1, an end-user (who had little prior-experience with Taverna) describing an external-reuse scenario for learning purposes]

4.5 Finding 5: When Composition Model is Misaligned with Computation Model, it Leads to Difficulties in Composition

Another observation that we realized through our interviews was that end-users struggled when visual composition vocabulary was different than the type of computational vocabulary common for that domain. For instance, in the case

of Kepler environment, Participant P8 (beginner) had trouble including code blocks. Further more, in the case of SimMan3G simulator, when Participant P5 (beginner) was forced to write events while thinking in terms of dataflow, computational model mismatch was sighted as an important concern.

Here are some statements collected during the interviews to demonstrate what problems end-users faced when they has a mismatch in computation model:

- "Without a worksheet it would be very difficult to fill in the values in the frames. We need to write down all our thoughts and all information organized..." [Participant P5, SimMan3G commenting on difficulty to directly compose frames]
- "Well, it's relatively easy if I have to just combine simple operations. But if I add a differential equation and Kepler thinks about integration of things so I have to rewrite differential equations as integrates, changing code and that would be difficult..." [Participant P8, Kepler commenting on difficulty arising from adding mathematical expressions and code]

5 Conclusions

End-User Programming is an activity that has been attributed to allowing end-users — people who are not professional software developers — to program computers. An important class of end-user programming is writing compositions using various domain-specific composition environments such as workflow tools, widget compositions and simulation software. We argue in this paper that while many of these environments may have different computation styles, and a diverse set of application domains, they often have a common set of problems. By getting a better understanding of quality problems for end-users, platforms developers can build better composition environments.

In this paper we describe our qualitative study that throws a light on how end-users use their composition environments, what kind of quality concerns end-user have, and how they can be provided better infrastructures. Furthermore, often the same composition environment could be used for disparate audience with a varying level of skills. A better understanding of the quality requirements could help in better targeting of composition environments.

We presented the findings of our qualitative study in the previous section. Besides the general findings that we discussed in previous section, here are some recommendations for composition environment developers to improve the quality of composition environments.

1. To be effective, composition environments must support all these phases of composition: (i) search and exploration (ii) reuse (iii) construction (iv) analysis (v) execution, and (vi) debugging.
2. Self reuse is a more preferred form of reuse for end-users. However, unless the platforms developers provide curated repositories and specialized compositions for scenarios, the likelihood of external reuse is typically low.

3. Beginners still need tutorials and samples (learning phase). Repositories of compositions are a good mechanism to shorten the learning curve.
4. There is no need to dumb down the vocabulary for end-users. End-users are not overly bothered by complex vocabulary as long as environments provide mechanisms to interact and learn the vocabulary.
5. There is a need for more analytic support across all composition environments. Naive users often require automated analyses to know "what is going on?" with their external reuse scenarios, while Experts need analyses to know "how to get it right?" to modify compositions and adapt them to a different setting in an internal reuse scenario.
6. While construction and execution may seem distinct activities, often execution phase is interleaved with the construction phase. To build high-quality composition environments, platform developers need to support iterative execution that is widely used both as a learning and debugging mechanism.
7. As much as it is possible, compositions should match the computation styles of the domain. A mismatch makes composition process hard and adds an additional burden on end-users to write error-free compositions.

Acknowledgments. This work is supported in part by the National Security Agency. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Security Agency or the U.S. government.

References

1. Deelman, E., Singh, G., Mei-Hui, S., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A.C., Jacob, J.C., Katz, D.S.: Pegasus: a framework for mapping complex scientific workflows onto distributed systems. Sci. Program. **13**(3), 219–237 (2005)
2. Dwivedi, V., Velasco-Elizondo, P., Maria Fernandes, J., Garlan, D., Schmerl, B.: An architectural approach to end user orchestrations. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 370–378. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23798-0_39
3. Edwards, R., Holland, J.: What is Qualitative Interviewing? The 'What is?'. Research Methods Series. Bloomsbury Academic (2013)
4. Garijo, D., Corcho, Ó., Gil, Y., Braskie, M.N., Hibar, D.P., Hua, X., Jahanshad, N., Thompson, P.M., Toga, A.W.: Workflow reuse in practice: a study of neuroimaging pipeline users. In: 10th IEEE International Conference on e-Science, eScience 2014, Sao Paulo, Brazil, 20–24 October 2014, pp. 239–246 (2014)
5. Garlan, D., Dwivedi, V., Ruchkin, I., Schmerl, B.R.: Foundations and tools for end-user architecting. In: Large-Scale Complex IT Systems. Development, Operation and Management - 17th Monterey Workshop, UK, pp. 157–182 (2012)
6. Goodell, H.: End-user computing. In: CHI 1997 Extended Abstracts on Human Factors in Computing Systems: Looking to the Future, CHI EA 1997, NY, USA, p. 132 (1997)
7. Ko, A.J., Abraham, R., Beckwith, L., Blackwell, A.F., Burnett, M.M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B.A., Rosson, M.B., Rothermel, G., Shaw, M., Wiedenbeck, S.: The state of the art in end-user software engineering. ACM Comput. Surv. **43**(3), 21 (2011)

8. Lee, C., Nordstedt, D., Helal, S.: Enabling smart spaces with osgi. IEEE Pervasive Comput. **2**, 89–94 (2003)
9. Letondal, C.: Participatory programming: Developing programmable bioinformatics tools for end-users. End-User Development, pp. 207–242 (2005)
10. McConahy, A.L., Herbsleb, J.D.: Platform design strategies: contrasting case studies of two audio production systems. In: FutureCSD Workshop at CSCW (2011)
11. Miles, M.B., Huberman, A.M., Saldaña, J.: Qualitative Data Analysis. SAGE Publications, Thousand Oaks (2013)
12. Moore, D.M., Crowe, P., Cloutier, R.: Driving major change: The balance between methods and people. Software Technology Support Center Hill AFB UT (2011)
13. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge (1993)
14. Saldana, J.: The Coding Manual for Qualitative Researchers. SAGE Publications, Thousand Oaks (2015)
15. Schmerl, B.R., Garlan, D., Dwivedi, V., Bigrigg, M.W., Carley, K.M.: SORASCS: a case study in SOA-based platform design for socio-cultural analysis. In: International Conference of Software Engineering (ICSE), pp. 643–652 (2011)
16. Segal, J.: Some problems of professional end user developers. In: VL/HCC, pp. 111–118 (2007)
17. Villa, F., Athanasiadis, I.N., Rizzoli, A.E.: Modelling with knowledge: a review of emerging semantic approaches to environmental modelling. Environ. Model Softw. **24**(5), 577–587 (2009)