Model Self-Assessment of Writing Principles
Principle 3: Agents, Actions, and Objects

Norman Ramsey

March 2015

Original and revised texts

The original and revised texts are shown in Figure 1. The original text is the opening paragraph of a paper I wrote in 1996, before I had begun to study any of the methods we are learning in class. The paper was sent to the conference on Programming Language Design and Implementation. The audience is interested in details of programming-language implementation, and they are very familiar with compilers, assemblers, and linkers.

Analysis and explanation

Here are the listed elements (a) to (f) of the analysis and explanation called for in the handout.

(a) The important agents, actions and objects in my paragraph are shown in Table 2.

(b) My goal in this paragraph is to leave my reader convinced that relocatable object code is necessary because it solves an important problem. I especially want to focus my reader's attention on the issue that when a compiler is translating source code, it doesn't yet know the exact locations that the names of code and data will eventually stand for.

In the context, the key agents are the ones needed to get from source code to executable binary: the compiler, assembler, and linker. Of the many agents they act on, I want to focus my reader's attention especially on the locations, but also on possible representations of compiled or assembled units (because relocatable object code is the key representation that this paper is about).

Finally, it's worth mentioning that I expect my audience (PLDI) to be very familiar with compilers, assemblers, linkers, and how they work. Not all readers will be familiar with the work cited in sentence 5, but the rest of it is all well known, and I am trying to remind people of familiar information, not to communicate new information.

(c) Subjects and verbs from the original text:

O1. Compiling is

O2. A compiler is able to emit

O3. Using assembly language

O4. Units can be linked

O5. It is believed

(d) Diagnoses:

O1. Neither "compiling is" nor "compiling speeds up" is well aligned with the agents and actions in my table. By splitting original sentence 1 into two sentences, I make it possible to shift the second part from "compiling speeds up" to "system compiles." This revision, although not perfect, is more closely aligned with the table in Figure 2. (The "system" is the composite agent made up of the compiler, assembler, and linker acting in concert.)

O3. "Using makes" was way off, and I've improved it to "the compiler can meet." This choice is discussed further below.

O4. Main verb "link" is not an important action. I've changed it to "compose," which is better aligned with my table.

O5. The main clause "it is believed" has nothing to do with any agent or action in my table. I've managed to move "it is believed" from the main clause into a subordinate clause, giving me the better aligned "an assembler can translate."

(e) Here are the grammatical subjects and verbs in the revised text:

R1. Compiling is

Original

O1 Compiling whole programs is slow; compiling units separately and linking the compiled units into a program speeds up the edit-compile-go cy-
O2 cle. For separate compilation, a compiler must be able to emit instructions and data without knowing the exact locations either of the instructions and data the compiler itself emits, or of the instructions and data emitted by other com-
O3 pilations. Using assembly language makes this task easy, because in assembly language all lo-
O4 cations are represented symbolically. Symbolic, assembly-like units can be linked to form programs (Fraser and Hanson 1982; Jones 1983), but the linker or loader must translate all units from symbolic form into the binary representation re-
O5 quired by the target hardware. It is believed to be more efficient to translate each unit separately into a binary form called *relocatable object code*.

Revised

R2 Compiling whole programs is slow. To speed up the edit-compile-go cycle, a typical programming system compiles one unit at a time, sep-
R3 arately, then composes units using a linker. In such a system, the compiler must emit instructions and data without knowing the exact locations either of the instructions and data the compiler itself emits, or of the instructions and data
R4 emitted by other compilations. The compiler can easily meet this requirement by emitting assembly language, which represents all locations sym-
R5 bolically. Symbolic representations of compiled units can be composed directly (Fraser and Hanson 1982; Jones 1983), but the linker or loader must translate the units from symbolic form into the binary representation required by the target
R6 hardware. As an alternative, which is believed to be more efficient, an assembler can translate each unit separately into a binary form called *relocatable object code*.

Figure 1: Original and revised texts

Agent	Action	Object
Compiler	emits	instructions and data
Compiler	doesn't know	exact locations
Compiler	emits	assembly language
Assembler	represents	locations
Assembler	hides	knowledge of locations
Assembler	emits	relocatable object code
Assembler	translates (to object code)	symbolic code
Linker	composes	units
Linker	composes	relocatable object code
Symbolic data	represents	units
Relocatable object code	represents	units

Figure 2: Important agents, actions, and objects

R2. System compiles

R3. Compiler must emit

R4. Compiler can meet

R5. Representations can be composed

R6. An assembler can translate

(f) Only sentences R3 and R6 respect the principle. Sentence R5 comes close to respecting the principle, but in order to improve information flow, it's been placed into the passive voice, with an object acted upon as the grammatical subject.

Sentence R1 is a statement of the problem. Because the essential problem is a property (slowness) rather than any action, I don't see how sentence R1 can ever conform to the principle.

Sentence R2 refers to a ghostly agent that doesn't really exist: the "system" formed by the combination of compiler, assembler, and linker. So the subject's not terrible. The main verb, "compiles," is not an action that I considered terribly important for purposes of this paragraph, but it's the background information for this problem. So subject and main verb are chosen from reasonable but secondary characters and actions.

In sentence R4, the subject is the compiler, which is a main player, so that's good. But the main verb is "meet," which is not at all reasonable. However, please notice the adverbial clause "by emitting assembly language," with its verb "emit." What happened here was that if I tried to use "emit" as the main verb, it destroyed the flow of information in the paragraph. By opening with "meet this requirement," I'm able to connect to old information from the previous sentence, and then I can add the two new pieces of information: emitting assembly language, and assembly language representing things symbolically.

If we go back and look at sentences R2 and R4 again, we'll see "composes units using a linker," "emitting assembly language," and "represents locations," all of which refer to key actions and objects, but in subordinate clauses rather than main clauses.

Self-review

Problems diagnosed in the revised text:

- As identified in part (f) above, there are several sentences in which main verbs don't match important actions; the important actions are relegated to verbs in subordinate or other clauses.
- I'm worried that maybe the paragraph is trying to do too much. Should it really deal with the work of Fraser and Hanson or Jones? Or should that material be relegated to another paragraph?

Level of mastery:

- Some parts of the text respect the principle, but for reasons stated above, I have chosen to disregard the principle in sentences R2, R4, R5, and R6. And in sentence R1, I was not able to find a way to respect the principle.

Summary judgement:

- The text says mostly what I want it to say, and I can't get it any better.

  I feel that I've shown a solid understanding of the principle, but in the majority of the sentences, I've either been unable to apply it or I've chosen not to apply it. For the work presented here, I would give myself a "not yet" assessment: the evidence of mastery is on the weak side, and I need to try again using a more suitable paragraph.

Results of peer review

My thanks to Caleb Malchik for reviewing the revised text. We agree broadly on which sentences follow the principle, although Caleb was unsure whether the agent and action "the assembler can translate" in sentence R6 are important enough to be considered to follow the principle.

Caleb also urged me to revisit sentence 3 and to consider whether "units" and the "finished program" should be made important characters. This comment shows pretty clearly that I haven't adequately communicated my ideas, because "units" are listed in my table as an important object.

Caleb also was not sure how relocatable object code differs from assembly, but I believe this distinction is already firmly fixed in the minds of my intended readers and so does not need to be further developed in the text.

I wish to revisit and revise the text in light of Caleb's comments, but I'm out of time.

Principle 3: Agents, Actions, and Objects

Norman Ramsey

Material for peer review

Context

The paragraph is the first paragraph of a paper sent to the conference on Programming Language Design and Implementation. The audience is interested in details of programming-language implementation, and they are very familiar with compilers, assemblers, and linkers.

Revised text

R1 Compiling whole programs is slow. To speed up
the edit-compile-go cycle, a typical programming
system compiles one unit at a time, separately,
R3 then composes units using a linker. In such a sys-
tem, the compiler must emit instructions and data
without knowing the exact locations either of the
instructions and data the compiler itself emits, or
of the instructions and data emitted by other com-
R4 pilations. The compiler can easily meet this re-
quirement by emitting assembly language, which
R5 represents all locations symbolically. Symbolic
representations of compiled units can be composed
directly (Fraser and Hanson 1982; Jones 1983), but
the linker or loader must translate the units from
symbolic form into the binary representation re-
R6 quired by the target hardware. As an alternative,
which is believed to be more efficient, an assembler
can translate each unit separately into a binary
form called *relocatable object code*.

Mastery statement and summary judgement

Level of mastery:

- Some parts of the text respect the principle, but for reasons enumerated in my analysis, I have chosen to disregard the principle in sentences R2, R4, R5, and R6. And in sentence R1 I was not able to find a way to respect the principle.

Summary judgement:

- The text says mostly what I want it to say, and I can't get it any better.

I feel that I've shown a solid understanding of the principle, but in the majority of the sentences, I've either been unable to apply it or I've chosen not to apply it. For the work presented here, I would give myself a "not yet" assessment: the evidence of mastery is on the weak side, and I need to try again using a more suitable paragraph.