Developer categorization of data structure fields (part 2 of 2)

Experiment performed at the 2008 ACCU Conference

First published in CVu vol. 22 no. ?

Derek M. Jones Knowledge Software Ltd derek@knosof.co.uk
David Binkley Loyola College binkley@cs.loyola.edu
Dawn J. Lawrie Loyola College lawrie@cs.loyola.edu

1 Introduction

This article is the second of two that investigates patterns in the organization of structure data types. The first part looked at patterns in a large number of **`struct`** definitions extracted from some large programs (e.g., Mozilla and Linux) written in C. This second part analyses the results of an experiment involving professional software engineers performed at the 2005 and 2008 ACCU conferences. This analysis of source code and this experiment are a first step towards understanding the decision making process developers go through when creating the data types used to represent information present within application software.

Experience shows that unless the number of solutions to a given problem is heavily constrained different people will often generate completely different solutions. For even the smallest collection of information there are many different data structures that could be used to represent it. One of the aims of the experiment described in this article is to investigate the extent to which different developers make similar decisions when creating data structures to represent the same information. Another aim is to investigate the impact that the specification has on the ordering of attributes (i.e., fields) in definitions.

The experiment performed at the two conferences asked subjects to create the data structures part of an API that could be used by programs to manipulate various items of information. The following is a list of some of the decisions that subjects need to make when creating this API:

- the items of information (i.e., the fields) contained in the same structure definition,
- the ordering of fields within a single data structure (e.g., a **`struct`** type).
- the name of a field and its type,
- the supporting scalar types that need to be defined. Examples include enumerated types or names given to scalar types (e.g., through use of **`typedef`**). While this is an interesting topic there was not sufficient experimental data to perform any meaningful analysis and it is not discussed further.

The issues discussed in the article are not usually covered by API coding guidelines, which tend to deal with issues such as naming conventions, error handling techniques, default parameters and which default constructs should be defined.[3]

For even a small API there is a huge number of combinations of possible structure type definitions. For instance, there are 2^{10} ways in which the 10 items of information could be split between two structure definitions, 3^{10} between three definitions and so on. The hypothesis that underlies this experiment is that subjects' decisions will be effected by their past experience (e.g., lifetime experience of interacting with the items appearing in the problems and programming experience creating and using data types) and the way in which information is presented in the API specification. The result of shared semantic influences is that patterns will exist in the definitions created by different subjects. This article attempts to uncover and offer explanations for some of these patterns.

2 Psychology studies

The general problem of how people go about the process of clustering items into categories (which is one way to think about how fields are organized into structure definitions) has been extensively studied by psychologists. A brief overview of some of the key findings from these studies is given in this section. To begin with, children as young as four have been found to use categorization to direct the inferences they make,[4] and many different studies have shown that people have an innate desire to create and use categories (people have also been found to be sensitive to the costs and benefits of using categories[11]). By dividing items in the world into categories of things, people reduce the amount of information they need to learn[15] by effectively building data structures that enable them to lookup information on an item they may not have encountered before, assigning that item to one or more categories, and extracting information common to previously encountered items in those categories. For example, which attributes of an item are used for lookup and how attributes are compared are some of the issues studied by psychologists. For instance, a

flying object with feathers and a beak might be assigned to the category *bird*, which suggests additional information such as it lays eggs and may be migratory.

Of necessity the description of items of information in a specification will name objects. To what extent do the names people use for objects effect the categories they are assigned to? For instance, in English the name given to a large stuffed seat for one person is the same as that for a wooden chair, while in Chinese such a large stuffed seat is given the same name as a large stuffed seat for two or more people. In a cross language study where the subjects were native speakers of either English, Chinese, or Spanish[12, 13] Malt, Sloman and Gennari showed subjects pictures of objects of various shapes and sizes that might be capable of belonging to either of the categories— bottle, jar, or container, the subjects were asked to name the objects and also to group them by physical qualities. The results found that while speakers of different languages showed substantially different patterns in naming the objects, they showed only small differences in their perception of the objects (i.e., the categories based on physical attributes).

2.1 Category formation

How categories should be defined and structured has been an ongoing debate within all sciences. For instance, the methods used to classify living organisms into family, genus, species, and subspecies has changed over the years (e.g., most recently acquiring a genetic basis). Culture (i.e., the society in which a person grew up) has also been found to influence this process[14]

What method do people use to decide which, if any, category a particular item is a member of? The following are some of the different theories that have been proposed (see the books listed in Further reading for more details):

- The defining-attribute theory proposes that members of a category are characterized by a set of defining attributes. This theory predicts that attributes should divide objects up into different concepts whose boundaries are well defined. All members of the concept are equally representative.
- The prototype theory proposes that categories have a central description, the prototype, that represents the set of attributes of the category. This set of attributes need not be necessary, or sufficient, to determine category membership. The members of a category can be arranged in a typicality gradient, representing the degree to which they represent a typical member of that category.
- The exemplar-based theory of classification proposes that specific instances, or *exemplars*, act as the prototypes against which other members are compared. Objects are grouped, relative to one another, based on some similarity metric. The exemplar-based theory differs from the prototype theory in that specific instances are the norm against which membership is decided. When asked to name particular members of a category, the attributes of the exemplars are used as cues to retrieve other objects having similar attributes.
- The explanation-based theory of classification proposes that there is an explanation for why categories have the members they do. For instance, the biblical classification of food into *clean* and *unclean* is roughly explained by saying that there should be a correlation between type of habitat, biological structure, and form of locomotion; creatures of the sea should have fins, scales, and swim (sharks and eels don't) and creatures of the land should have four legs (ostriches don't). From a predictive point of view, explanation-based categories suffer from the problem that they may heavily depend on the knowledge and beliefs of the person who formed the category; for instance, the set of objects a person would remove from their home while it was on fire.

3 Threats to validity

For the results of this experiment to have some applicability to actual developer performance it is important that subjects work through problems at a rate similar to that which they would process source code in a work environment. Subjects were told that they are not in a race and that they should work at the rate at which they would normally process code. However, developers are often competitive and experience from previous

experiments has shown that some subjects ignore the work rate instruction and attempt to answer all of the problems in the time available. To deter such behavior during this experiment the problem pack contained significantly more problems than subjects were likely to be able to answer in the available time (only one person got as far as attempting the last problem).

Developers often have the opportunity to interact with (e.g., ask questions of) people including the following: domain experts who wrote the specification, likely users of the API, and the paying customer. There was no such opportunity in this experiment; thus, subjects only had their own interpretation of the specification with which to work.

Developers often have the opportunity to study how information is going to be accessed by applications. This allows the data structures to be optimized for common access patterns or to make writing the code simpler (e.g., by reducing the number of arguments that might need to be passed). These design decisions are often made via an iterative process that may involve interaction with the code that uses them.

The creation of data structures is usually an iterative process and the answers provided by subjects in this experiment can only be regarded as a first iteration. Each iteration requires a lot of time and it is not known if subjects considered it more important to answer multiple problems than provide a more refined answer to a smaller number of problems.

3.1 Subject experience

Traditionally, developer experience is measured in number of years of employment. In practise the number of lines of source code read and written (interaction with source code overwhelmingly occurs in its written, rather than spoken, form) is a more accurate measure of source code experience. Developer interaction with source code is rarely a social activity (a social situation does occur during code reviews), and the time spent on these social activities is probably small enough to ignored. The problem with the read/write measure is that it is very difficult to obtain reliable estimates of the amount of source read and written by developers. This issue was addressed in studies performed at previous ACCU conferences.[7–9]

What is the relationship between lines of source code written by a developer and the number of data structures they have created? Modifying existing code is a common developer activity. This is likely to result in existing data structures being modified rather new ones created. Your authors do not know of any empirical data that shed light on the relationship between lines of code written and data structure created.

A persons background knowledge has been found to effect the categories they create.[6] Experience shows that it is not unusual for the creator of an API to be a non-expert in the domain being addressed. For consistency it would be advantageous if all subjects taking part in the experiment were either domain experts or not domain experts. No information on subject experience with the domains used in the problems is available.

3.2 Subject motivation

Subjects are presented with a list of various kinds of information and have to rely on their experience to design data structures to be used by programs that process this information. Subjects have to estimate the trade-offs among a large number of possible different structure definitions that could be created. At the two extremes are a single structure containing all of the information could be created and separate structure definitions for each piece of information. The following are some of the factors that might play a significant part in the trade off process:

- Simplicity. Writing code is simplified when all required information is obtained by accessing a single structure.
- Information hiding. It may be necessary to restrict access to some of the information, perhaps for commercial or security reasons, or to reduce the likelihood that a fault in one part of a program can effect unrelated data in another part. Distributing the information across different structures is a means of organizing code that helps to minimise the information available at given points in a program.
- Minimising the impact of future updates, with the aim of making it easier to change one part of a system without having to update unrelated parts. For instance, if all of the information was held in

a single structure then when a new field is added or an existing field changed, all of the code that referenced the structure would need to be recompiled. If the information was organized into separate structures based on semantic relatedness then adding a new field or changing an existing field, means that only code referencing that structure needs to be investigated when considering what modifications need to be made.

- Minimising the amount of storage used during program execution. For instance, consider a tree whose nodes are represented by a large single structure where each node contains some fields representing frequently accessed information and other fields representing seldom accessed information. In this case, dividing the single structure into multiple structures can yield considerable storage and performance savings,[1] with frequently accessed fields allocated to one structure and the less frequently accessed information in different structures.

4 Experimental setup

The experiment was run by one of your authors during a 40 minute lunch time session at each of the 2005 and 2008 ACCU conference (http://www.accu.org) held in Oxford, UK; between 250 and 300 professional developers attend this conference every year. Subjects were given a brief introduction to the experiment, during which they filled in background information about themselves, and they then spent 30 minutes (2005) and 20 minutes (2008) working on the problems. All subjects volunteered their time and were anonymous.

4.1 The Problem

The following is an excerpt of the text instructions given to subjects:

You will be asked to define some data structures to hold values for various kinds of information. This is not a race and there are no prizes for providing answers to all questions. If you do complete all the questions feel free to add any additional comments to your previous answers.

Yumei is a medium sized island that gained independence in 1945. The island economy is based on the export of a various kinds of raw materials and agricultural products. The government has decided to publish a set of APIs that applications written for government departments must follow.

The government could not reach consensus on an object oriented approach and it was decided to specify data types and access functions separately. You have been assigned the job of creating the data structures that will be used to hold various kinds of information. If possible please briefly specify any rationale used to decide how to organise the data structures.

The declarations can be written in a language of your choice (it would simplify subsequent analysis if either C or Java were used). The members may have any arithmetic type, or a pointer type, or an array type.

1. Read the information that needs to be represented.
2. Decide which information should be held in a given data structure.
3. Decide on a type to be used to represent the value of the information.
4. Decide on the name of a member to be used to denote the information.
5. Define one or more structure types to hold members denoting the information.

The following is an example question

The Department of Housing maintains rented accommodation for workers. The form of accommodation can be one of:

```
Bungalow
Caravan
Dormitory
Flat
House
Youth hostel
```

The information, about the accommodation, that needs to be processed by applications includes the following (in alphabetical order):

1. Address of accommodation
2. Detached, Semi-detached, Terraced
3. Easily moved
4. Floor number of accommodation
5. Name of current responsible occupant
6. Name of given town
7. Number of bathrooms
8. Number of bedrooms
9. Number of each kind of accommodation in given town
10. Number of rooms
11. Parking space included with accommodation
12. Shared bathroom
13. Size of garden
14. State of repair
15. Total area of floor space
16. Total value of each kind accommodation in given town
17. Weekly rent

```
#define HOUSE          0
#define BUNGALOW       1
#define CARAVAN        2
#define FLAT           3
#define DORMITORY      4
#define YOUTH_HOSTEL   5
#define MAX_ACCOM_KIND 6

#define NOT_APPLICABLE 0
#define DETACHED       1
#define SEMI_DETACHED  2
#define TERRACED       3

struct town_accommodation {
            char * town_name;
            int num_accom_kind[MAX_ACCOM_KIND];
            float value_accom_kind[MAX_ACCOM_KIND];
            };

struct accommodation {
            int kind_of_accommodation;
            int num_rooms;
            int num_bedrooms;
            int num_bathrooms;
            float garden_size;
            float weekly_rent;
            int detached_status;
            char easily_moved;
            char shared_bathroom;
```

```
            char has_parking_space;
            int floor_number;
            char * address;
            char * repair_state;
            char * current_occupants;
            };

typedef unsigned char BOOL;

struct caravan_rec {
            int number_of_rooms;
            int number_of_bedrooms;
            int number_of_bathrooms;
            float size_of_garden;
            float rent_per_week;
            BOOL bathroom_is_shared;
            BOOL parking_space_available;
            int floor_number;
            char * address;
            char * state_of_repair;
            char * current_occupiers;
            };

struct accommodation_rec {
            int accommodation_kind;
            int number_of_rooms;
            int number_of_bedrooms;
            int number_of_bathrooms;
            float size_of_garden;
            float rent_per_week;
            int detached_status;
            BOOL bathroom_is_shared;
            BOOL parking_space_available;
            int floor_number;
            char * address;
            char * state_of_repair;
            char * current_occupiers;
            };
```

End of excerpt of text instructions given to subjects.

The experiment brochure handed out to subjects briefly specified three different kinds of domain specific information, each of which followed the format given in the Example:

- Department of Agriculture. This involved: Meat (Cattle, sheep, and pigs) and Cereals (barley, corn and wheat) and the following list of items: Owner of Farm, Farm acres used to grow crops, Seed supplier, Location of field, Fertilizer costs, Weed killer sprayed, Fertilizer applied, Date crop harvested, Weight of average shipment, Was organically produced, Number of each kind of animal on farm, Acre of crop market value, Farm location, Antibiotics used, Feed-stuff costs, Market value of kind of animal, Genetic ID, Crop sown date, Last vet inspection date, Average labor cost for growing crop, Farm acres used to rear animals, Pedigree ID, Date animal slaughtered, Date animal born and Animal raising average labor cost. Some subjects saw a subset containing 15 items, where some items had been combined into a more generic description (e.g., “Crop sown date” and “Date animal born” becoming “Date agricultural product started life”).

- Department of Transport. This involved: Self driven (Walking, Cycling, Car and Motor bike) and Passenger (Ferry, Train, Bus, and Flying) modes of transport and the following list of items: Arrival time affected by congestion, Weekly travel cost, Method of transport is environment friendly, Travel distance, Yearly expenditure on infrastructure supporting transport method, Workforce percentage

using transport method, Travel time, Tax revenue from transport method and Probability of arriving on time.

- Department of Natural Resources. This involved: Industrial stone (Marble, Slate, and Granite) and Purified metal (Gold, Silver, and Titanium) and the following list of items: Raw material per shipment, average weight, Production volume per week, Mining location, Raw material value on commodity exchange, Storage cost per week, Cost of extraction per unit weight, Mining date of raw material and Dispatch method (e.g. sea/air).

4.2 Variations on the problem

There are various ways in which the problem information seen by subjects could be organized. The format of the experiment meant that the organization had to be kept as simple as possible. The following describes the information organization that was varied:

- The order in which items of information were listed in the problem specification. For instance, in the Example problem the specification uses the order "Name of given town" followed by "Number of bathrooms". The order in which items were listed was randomly chosen for each printed set of problems.
- In many cases it is possible to use different phrases to describe an item of information. For instance, the phrase "Parking space included with accommodation" can also be worded as "Accommodation has parking space included with it". In most cases the phrase seen by each subject was randomly selected from two possibilities (in a few cases only one phrase was reasonable).

5 Results

Approximately 240 (2005) and 300 (2008) people attended the conference, of which 11 (2005) and 15 (2008) took part in the experiment. The 26 subjects produced 38 answers (Agriculture: 22 Transport: 12 Natural Resources: 1. (Three subjects gave an answer to the example problem.) Some answers were incomplete at the end of the allotted time. On average each participant completed 1.5 problems. The average number of years of experience of subjects was 11.1 (standard deviation 6.3). Subjects were told that they could use any computer language and the languages used were (subject counts in parenthesis) C++ (15), C (8), Java (1), C# (1) and Python (1).

For the Agriculture problem, the average number of structure definitions created per answer was 5.2 (standard deviation 2.5), and these definitions contained on average of 5 fields (sd 3.5) (this includes fields not found in the *specification*).

In the following discussion of the study's results the term *specification* is used to refer to the description of an item as found in the question seen by a subject (e.g., "Size of garden"), while the term *field* will be used to denote the name given by the subject (e.g., "`garden_size`"). The term *component* is used to refer to a word or abbreviation found within a field. *Components* might be separated using underscores or camel casing. For example, "`garden_size`" includes two components: "`garden`" and "`size`".

Subjects sometimes wrote comments on their answer sheets that were obviously intended to be directed at those running the experiment rather than source code comments that were part of an answer to the problem. The common comment was that there was insufficient time to complete an answer.

5.1 Influence of specification on information ordering in declarations

The information in a specification is ordered in various ways and it is hypothesized that this ordering has an effect on how developers order various entities within a definition. The following are the information ordering issues analysed in this subsection:

- the effect that the order in which items of information are listed in a specification has on the order in which fields are defined in a structure. For instance, if the specifications lists the "number of bathrooms"

before the "number of bedrooms" the field `number_of_bathrooms` may be expected to appear before `number_of_bedrooms` (assuming they both appear in the same definition and that these names are used).

- the effect that the words used and their ordering in each specification has on the components of a field. For instance, the specification "number of bedrooms" might be expected to result in the field `number_of_bedrooms` while the specification "bedroom count" might be expected to result in the field `bedroom_count`.

Subjects sometimes gave answers that did not contain some items of information. This is likely to have been an oversight, although at least one subject made an explicit decision (their answer contained a comment explaining how some items could be deduced from information on other items, e.g., *organically produced* from information on the fertilizer/weedkiller used).
Some subjects gave answers that included fields that did not directly correspond to an item listed in the specification (e.g., minimum/maximum/mean/standard deviation arrival delays in the Transportation API). Such fields were ignored in the analysis.

5.1.1 Item words in field names

Identifiers are often formed by combining two or more known words. If subjects are influenced by the content of a specification it is to be expected that the words used and the order in which they occur will be reflected in the name of the corresponding field. For instance, given the description "Fertiliser costs" a field name of `fertiliser_costs` or `fertiliser_cost` might be expected, while the specification "Cost of fertiliser" might result in the field name `cost_of_fertiliser` (perhaps with the *of* omitted) or `fertiliser_cost`.
It is possible that subjects have relatively fixed naming habits or that the given specifications induced subjects to use a particular naming. The results were therefore analysed by subject and by item of information as follows:

1. The name of a field is broken down into its components (e.g., `bathrooms_count` has the components `bathrooms` and `count` as does `bathroomsCount`). The algorithm is the same as that used in a previous ACCU experiment.[9]
2. For each component of a field what was not a dictionary word, the component was treated as an abbreviation and replaced to the appropriate dictionary word. Based on the specification, there was always a unique replacement of each abbreviation. For example, `num` was replaced with `number`.
3. The components from each field were compared to the words from the corresponding specification. A component and a word match if they have the save *stem* (e.g., the stem of "swimming" is "swim"). Stemming was performed using the Krovetz stemming algorithm.[10] This comparison was carried out twice: once ignoring the order of the component and word and the second time requiring the same relative order.

Counting the number of components that appear in the corresponding specification, in almost 70% of the cases all the components of a field occur in the specification of the corresponding item. Figure .1 gives a break down by fields containing 1, 2, 3 or 4 components and the percentage where the corresponding specification does not contain zero or more of these components.
The 50 cases, out of a total of 198 unique components, where a component did not appear in the corresponding specification can be broadly broken down into the following categories:

- the component is more specific: there may be several fields each relating to some aspect of the specification, e.g., an address specification includes information on street, postal-code, etc.). Another example is the fields `garden_length` and `garden_width` created to hold information on "Size of garden".

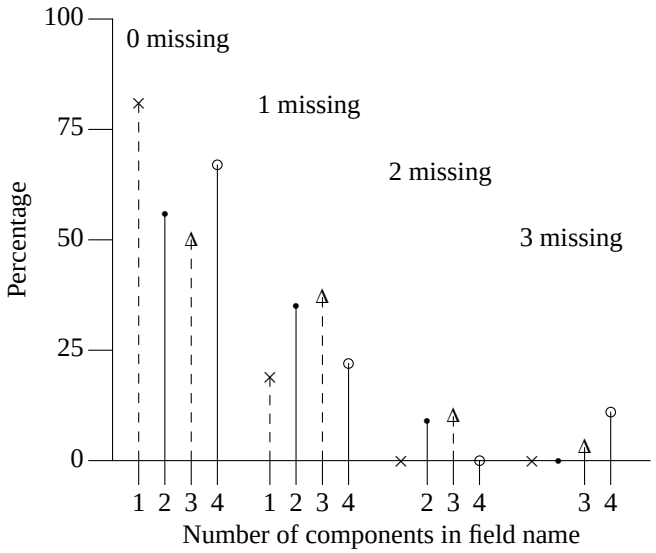

Figure .1: Percentage of fields containing a given number of components (e.g., 1, 2, 3 or 4) where the field's components also occur in the corresponding specification. In the group of four on the left all components occur in the specification; in the next group one component does not occur in the specification; and so on. Combined data from 2005 and 2008, a total of 260 fields having 704 components.

- synonym-like usage: for example *type* used in place of *kind* and *count* used in place of *number*, leading to the choice of the field name `product_type_count` for "Number of each kind of agricultural product".
- grammar support words: words such as *is* in `is_organic` and *of* in `cost_of_feed` are used to generate short phrases.

Having shown that a large percentage of components occur in the specification we now compare the relative ordering of components and the words of the specification. This ordering might be effected by the following.

- An existing set of rules that the subject always uses to create fields from combinations of words.
- The relative word order present in the corresponding specification.
- A desire for consistency in all field names. The specifications seen by some subjects have characteristics that require decisions to be made about conflicting choices.

The description of many items was randomly chosen from a small set of possibilities; thus in some cases the descriptions given for two semantically related specifications use different word order patterns. For instance, a subject may see the two specifications "Weedkiller costs" and "Cost of fertiliser" (rather than "Fertiliser costs" in the later case).

Subjects who base their choice of field name purely on the wording that appears in the corresponding specification will not pay any attention to the word order found in the specification. However, subjects trying to achieve a degree of consistency across semantically related fields may select one kind of word ordering over another, perhaps the word order from the description that is first encountered (likely the first one listed in the list of items).

Figure .2 breaks down the fields that contain 2, 3 or 4 components based on the number of components occurring in the same order as the corresponding words from the corresponding specification (a total of 260 fields). The figure omits fields having only one component in common and also omits the single data point of one field with 6 components (because patterns cannot be established from a single data point). There were no fields with 5 components.

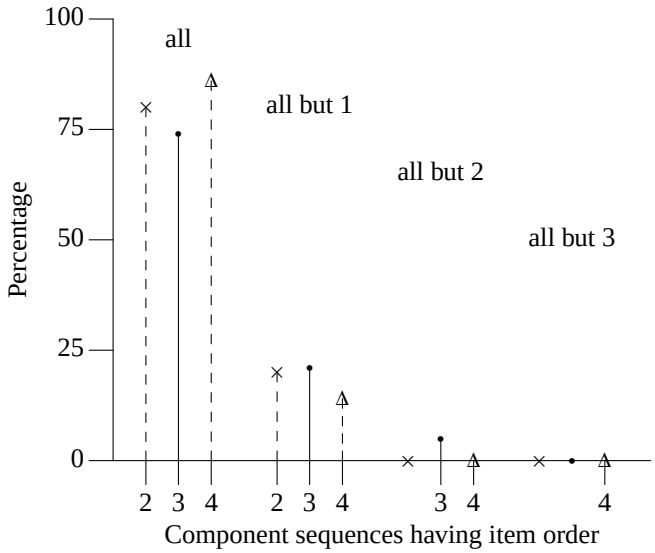

Figure .2: Percentage of fields containing a given number of components (e.g., 2, 3 or 4) where the field's components occur in the same order as in the corresponding specification (other words may appear in the specification between matching components). In the group of three on the left fields containing 2, 3 and 4 components all occurring in the same order as the corresponding specification, in the next group of one component does not occur in the same order and so on. Combined data from 2005 and 2008, a total of 260 fields.

Of these 260 fields, the components occur in the same order as the words of the corresponding specification in 86% of cases(37 cases did not follow this order). This provides strong evidence for the hypothesis that the subjects use the ordering of information in the specification when creating field names.
Table .1 lists some examples of fields definitions given as answers to a given specification. The samples are grouped under four common patterns that are apparent in the data.

Table .1: Some wording patterns and examples of them that appeared in subject answers. The "consistency with previous field" pattern occurs when the name of a field uses the same component ordering as that used by the field that occurred before it in the structure definition.

	Field	Specification
type name first		
	SCurrency currencyRevenue	Transport method tax revenue
	STime timeTravel	Travel time
favor preposition		
	float costPerWeek	Cost of travel per week
	float probabiltiyOnTime	Probability of arriving on time
	float revenueOfTax	Tax revenue from transport method
	float percentageOfUsers	Workforce percentage using transport method
consistency with previous field		
	int costs_material	Costs (e.g., Feed-stuff and Fertilizer) for agricultural product growing
	int costs_labor	Agricultural product average labor costs
noun adjective		
	boolean bathroomShared	Shared bathroom
	boolean parkingSpaceIncluded	Parking space included with accommodation
	String occupantName	Name of current responsible occupant

Is there a pattern to the 14% of cases where the specification order was not following? One possible pattern

is a preference for what might be considered to be *natural word order*. There are many patterns to English word order, with some orders being much more common than others. Perhaps subjects made use of their knowledge of what they consider to be *natural* word order. To test for this pattern of usage Google's n-gram dataset[2] was used; this data set includes counts of word pairs, triples, etc. as found by Google on the web.

The frequency of occurrence of components in each field and the corresponding specification word order were looked up in the Google dataset. For example, the components `start date` occur 872,156 times, while the words of the corresponding specification "date started" occur only 4,690 times. Thus the order of the components used by the subject in this answer is 186 times more common, in the Google observations, than the ordering used in the specification.

Table .2: Categorizes the fields components that appear in to different ordering based on observations in the Google n-gram dataset.[2] *No observations* - Number of field component sequences that did not appear in the Google n-gram database. *Subject more common* - The sequence used by the subject was more common than the specification. *About the same* - both sequences had similar number of occurrences, and *Specification more common* - the specification was the more common case.

	Count	Percentage
No observations	12	32.5
Subject more common	13	35.0
About the same	5	13.5
Specification more common	7	19.0

A breakdown of the 37 data points is presented in Table .2. In the table "No observation" include the 32.5% cases in which neither the components of the participants field nor the words of the specification occurred in the Google dataset. The row "Subject more common" represents cases in which the component order was at least 60% more common in the Google dataset that the specification word order. Similarly, "Specification more common" represents cases in which the specification word order was at least 60% more common in the Google dataset that the component order. The final category "About the same" includes the remaining cases.

The third of the data for which no observations were found all have three or more components that represent phrases rarely seen in the Google dataset, for example, "life start date".

At 35% use of the more common order was almost twice as likely as using an order that was less common than that appearing in the specification. Possible reasons for this usage is a desire by subjects for consistency of component order for related fields in the same structure definition.

5.1.2 Ordering of fields

Does the order of fields in structure definitions tend to follow the order in which the corresponding item of information appears in the specification? For instance, in the Example problem given earlier four items had the relative order: *Number of bathrooms*, *Number of bedrooms*, *Number of rooms* and *Shared bathroom*, and their corresponding fields had the relative order `num_rooms`, `num_bedrooms`, `num_bathrooms` and `shared_bathroom` (the two lists almost have opposite orders).

There are 24 possible ways this information can be represented using four fields (see Table .3). In a random selection every field ordering has a 1 in 24 probability of occurring, but what is the probability of a random field ordering matching that of the corresponding words listed in the specification?

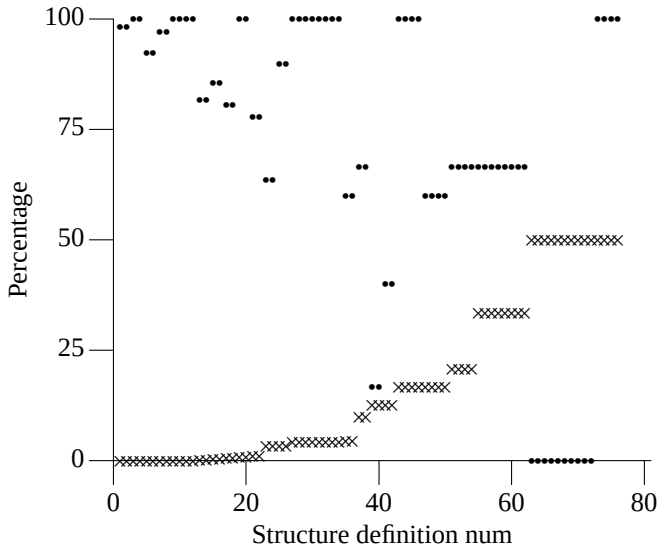

Figure .3: Probability that the measured field-item distance, for a given structure definition, could occur through random selection of fields (crosses) and the same structure's field-item distance expressed as a percentage of the maximum possible (bullets). If field order selection was random the bullets would be distributed around 50%. The structures measured are ordered, left to right, by increasing probability of field-item distance.

Table .3: The 24 possible ways in which four fields can be ordered. Rows have the same *field-item agreement metric* (the extent to which a field ordering matches that of the corresponding items in the specification, the higher the better).

Metric	Occurrences	Orders							
6	1	1234							
5	3	1324	2134	1243					
4	5	2314	3124	1342	2143	1423			
3	6	3214		2341	3142	1432	2413	4123	
2	5			3241		2431	3412	4132	4213
1	3					4321		4231	4312
0	1							4321	

To better understand this question, we count the *field-item agreement* for a sequence of fields. This metric represents the number of pairs that are in the expected order. For instance, the sequence 2314 is assigned the value 4: 2 appears before 3 and 4, 3 appears before 4 and 1 appears before 4. There are five sequences having an agreement metric of 4, see Table .3, and the probability that a random sequence of four fields will have an agreement metric of four is $5/24$.

It was noted that the sequence $1, 3, 5, 6, 5, 3, 1$ (second column in Table .3) is part of the Triangle of Mohanian numbers (sequence A008302 in the Encyclopedia of Integer Sequences (www.research.att.com/~njas/sequences) and also appears in the number of inversions of a permutation. This information was used to derive the possible number of structure definitions having a given field-item agreement metric for definitions containing larger numbers of fields.

Figure .3 shows the order distributions for the random case and the collected data. In the figure crosses at 50% indicate that there is a 50/50 change of the fields having the order they do (i.e., there are two fields in the definition). Moving left the crosses at 33.3% denote combinations involving three fields (there are also crosses at 16.67% for less likely combinations of three fields), crosses at 20.8% for some combinations of four fields, and so on. The crosses very close to zero are for definitions containing the less likely combinations of 6 or more fields, or just containing lots of fields.

If field order selection was random then on average the field-item distance would be 50% of the maximum

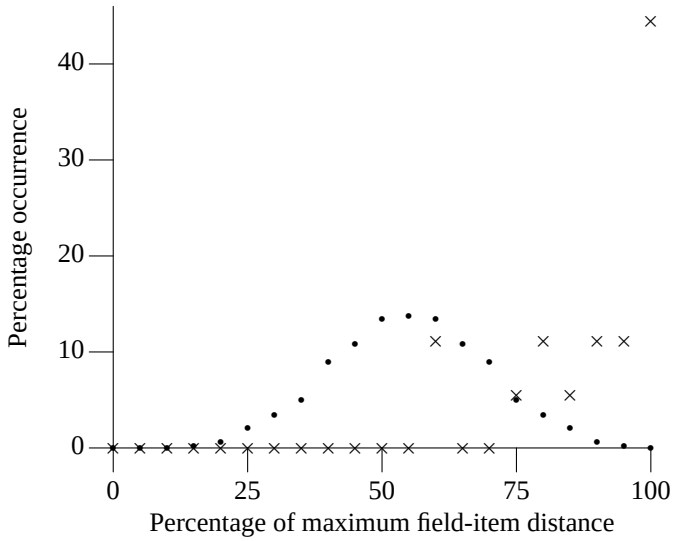

Figure .4: Bullets represent the probability (y-axis) of fields in a structure definition having a given percentage of the maximum field-item distance (x_axis). Crosses denote structure definitions in problems answers whose field-order is less that 5% likely to have randomly occurred. Counts have been put into bins of width 5 (percent).

value. The bullets in Figure .4 show the expected distribution of field-order distance that would occur with random field ordering. The crosses denote the percentage occurrence found in problem answers (answers with greater than 5% chance of occurring randomly were ignored). A significant percentage (44.4%) had the maximum field-item distance (i.e., the relative field order was the same as the items in the specification).

5.2 Information clustering

There are a lot of different ways in which information can be clustered into different structures. For instance, there are 2^{17} ways in which the 17 values listed in the earlier example could be split between two structure definitions, 3^{17} between three definitions and so on.

The problems subjects were asked to solve have no correct answer as such. If the development history for a variety of different APIs was available such things as development cost, ease of maintenance and the other factors discussed in the subsection on Subject motivation could be compared and those with lower costs and/or greater benefits labelled as being 'better' than the others.

The analysis in this subsection looks for general patterns in how items tended to appear together within the same structure definition.

5.2.1 Finding clusters

The organization of the data (i.e., a list of experimenter provide items clustered into separate structure definitions in a manner decided by each subject) and the analysis required (i.e., averaged over all subjects what clustering of items exists, and how strong is it) is very similar to a study performed Ross and Murphy[16] and the mathematical techniques used in that study are used here. In one of the experiments performed by Ross and Murphy subjects were given a list of various foods and asked to divide them into groups; either "similar food types", "foods that are eaten in the same situation" or "things that go together".

Ross and Murphy generated a Robinson matrix from the subject data and this was used to obtain information on the general form of the categories created by their subjects. A Robinson matrix has the property that the value of its matrix elements decreases, or stays the same, when moving away from the major diagonal (more mathematical details in the Ross and Murphy article[16]). A matrix can be put into this form by reordering its rows and columns. This reordering brings together entries that are similar and moves dissimilar ones apart. For this study the seriation package[5] within the R statistical program, `http://www.r-project.org`, was used to create a Robinson matrix. Figure .5 and Figure .6 are visual representations of this matrix for one

problem.
The *Department of Agriculture* problem produced the most answers (22); it also contained the largest number of items (25), making it much more likely that answers contain many structure definitions. The analysis in this subsection is applied to the answers to this problem.
The similarity detection process involves the following two stage process, for the answers to the Department of Agriculture API:

1. creating a similarity matrix from the structure specifications. The rows and columns of this matrix both represent the specifications listed in the problem, for instance if the entry for "Easily moved" is in row 3, then column 3 also denotes "Easily moved". The entries for this similarity matrix are calculated as follows: 1) Zero the matrix, 2) for each subject perform the following: a) add 1 to every entry in the matrix where two specifications occur together in the same structure definition (e.g., if a definition contains fields representing both "Easily moved" and "Shared bathroom" add 1 to the entry at the corresponding row/column (the matrix is symmetrical, so two entries will be incremented).
2. Within the seriation package the `seriate` function is capable of processing data having a variety of forms and implements a variety of algorithms that attempt to optimize the process of generating a Robinson matrix from this data (the computational cost is so high that approximations have to be made). It was found that consistent results were obtained by mapping the data to a dissimilarity form. This was done using the formula $1 - max\_element/max\_element\_val$ for each element of the matrix, where $max\_element\_val$ is the largest value in any element of the matrix.

The R instructions used to generate these images were:

```
library("seriation")
fmat <- as.matrix(read.table("f.data"))
fdist <- as.dist(1 - fmax/max(fmat))
fser <- seriate(fdist, method="BBURCG")
pimage(fdist, fser)
```

The items corresponding to the ticks along both axis in Figure .5 and Figure .6 are as follows (the alternative specifications are also listed):

```
Date of last vet inspection | Last vet inspection date
Date animal born | Animal birth date
Date crop sown | Crop sown date
Date animal slaughtered | Animal slaughter date
Date crop harvested | Crop harvest date
Antibiotics used | Antibiotics given
Fertilizer used | Fertilizer applied
Weed killer used | Weed killer sprayed
Pedigree ID | Pedigree class
Genetic ID | Genetic family
Supplier of seeds | Seed supplier
Was organically produced | Produced organically
Number of each kind of animal on farm
Farm acres used to grow crops | Acres used to grow crops
Farm acres used to rear animals | Acres used to rear animals
Average shipment weight | Weight of average shipment
Market value of kind of animal
Market value of acre of crop | Acre of crop market value
Average labor cost for raising kind of animal | Animal raising average labor cost
Average labor cost for growing crop | Crop growing average labor costs
Feed-stuff costs | Cost of feed-stuff
Fertilizer costs | Cost of fertilizer
Location of field | Field location
Location of farm | Farm location
Farm owner | Owner of farm
```

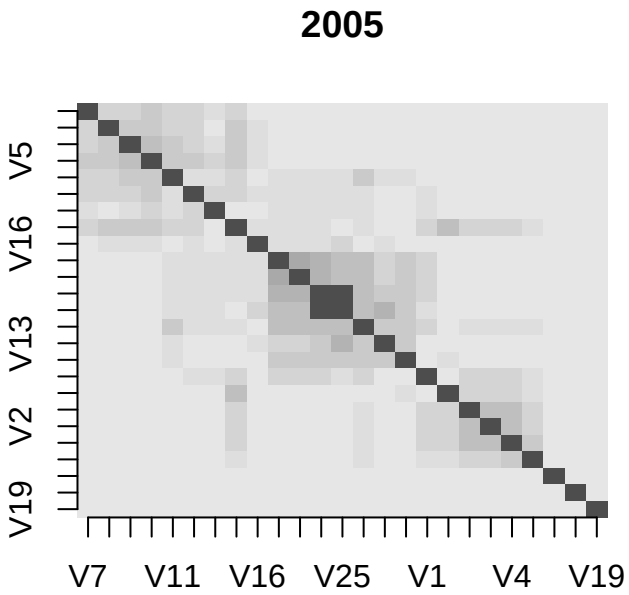

Figure .5: A visualization of the contents of the Robinson matrix generated from the 2005 field-item structure membership information. Darker areas indicate more closely associated fields. The order of items along the x and y axis is: 7, 3, 8, 5, 11, 22, 23, 12, 16, 15, 14, 24, 25, 21, 13, 17, 1, 10, 9, 2, 4, 6, 18, 20 and 19 (plot shows a subset of these values).

Comparing the clusterings extracted from the 2005 and 2008 answers (Figure .5 and Figure .6) it can be seen that both the 2005 and 2008 data appear to be made up of approximately three large clusters (combining the data from both experiments produces a result that is almost identical to that for 2008). Within each of these large clusters are smaller clusterings of field items.

The only two items that are consistently placed in the same definition are items 24 & 25 ("Location of farm" and "Farm owner") and items 14, 15 and 13 are often included in the same definition with them (the name given to definition containing this set of fields was often `farm` or something very close to this name; no analysis of this kind of type name was made). Other items are often placed in the a definition together the items 2, 4 and 6; other small sets of items can be extracted from the figures.

5.3 Comparison with source code measurements

The first part of this article included various measurements of a large amount of C source code. This subsection compares those measurements against the same measurements for the structure definitions given in the problem answers. The number of fields contained in the source code **`struct`** definitions varied between 2 and 70 fields, while the definitions given in the answers to the experiment problems contained a small range of fields (between 2 and 15; average 5).

The first part of the article included three figures containing measurements of field attributes and these are discussed in the following:

1. See Figure .7 for a comparison of experiment answers and source code measurements. For those deltas having a zero measured percentage there were no instances of a minimal type change sequence because of the small number of occurrences of a given type sequence (often only one occurrence).

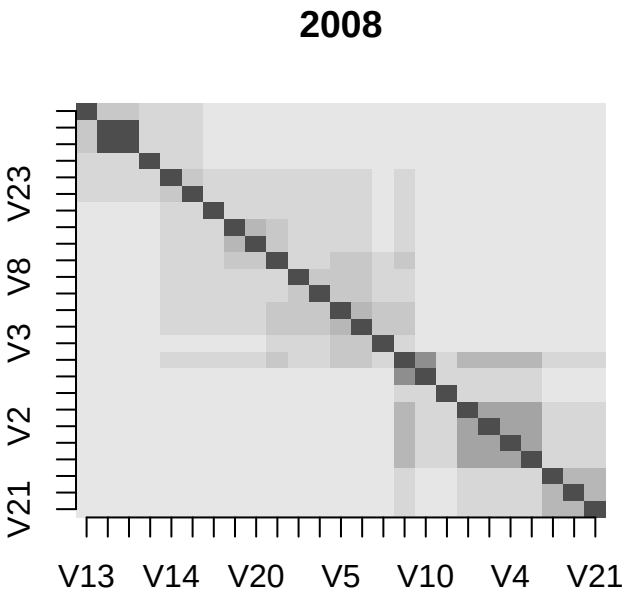

Figure .6: A visualization of the contents of the Robinson matrix generated from the 2008 field-item structure membership information. Darker areas indicate more closely associated fields. The order of items along the x and y axis is: 13, 24, 25, 15, 14, 23, 22, 18, 20, 11, 8, 16, 5, 7, 3, 12, 10, 9, 1, 2, 4, 6, 17, 19 and 21 (plot shows a subset of these values).

2. In the case of the number of individual field pairs that share one or more subcomponents because of the small number (i.e., 104) of structure definitions available for analysis in the answers there is a great deal of variation the number of occurrences of field-pair distances and it is not possible to make reliable comparisons against the source code measurements.
3. For the problem answers the average distance between field pairs for definitions containing a given number of fields appears to be similar to that found in the source code measurements. The small number of data points means it is not possible to make a stronger statement. For those wanting a comparison against the corresponding figure in part 1 of this article, the actual x/y axis values are: 3.14:5, 1.90:6, 1.00:7, 0.00:8, 1.00:9, 3.67:10, 0.00:11, 0.00:12 and 6.89:13.

6 Conclusion

This study investigates factors that influence the decisions made by developers when deriving API structure definitions from a specification.

The hypothesis was that various kinds of information in the specification had a significant impact on the way in which structure definitions were organized and the naming of fields. The particular attributes found to have an effect included:

- The ordering of fields was strongly influenced by the order in which items of information occurred in the specification.
- Any words appearing in the components of a field name were likely to also occur in the corresponding specification.

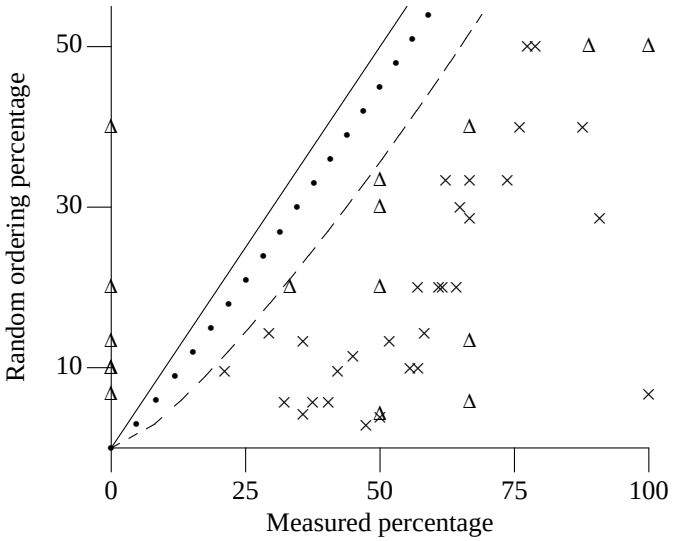

Figure .7: The measured percentage of structure definitions having a minimal type change sequence (x-axis) and the percentage that are expected to occur if fields were ordered randomly. A delta symbol indicates a problem answer percentage and the random percentage for a definition containing that particular number of types for problem answers, while cross applies to source code measurements. If field ordering were random the deltas/crosses would be expected to cluster along the solid diagonal line (bullets indicate 1 standard deviation, dashed line 3 standard deviations). Data for all definitions containing between four and seven (inclusive) fields.

- Any words appearing in the components of a field name were likely to either appear in the same order as that in the specification or to follow common English word order.

An analysis of individual structure definitions showed that for some items of information subjects made similar decisions on putting them together within the same structure definition. It is not known whether this clustering would have been stronger if subjects had had time to iterate on their initial design decisions.

Further experiments are needed to confirm the results of this study and to measure the consequences of allowing subjects more time to individual answer questions.

6.1 Further reading

A readable upper graduate level book dealing with some of the more higher level aspects of how people create and use categories "The Big Book of Concepts"" by Gregory L. Murphy published by MIT Press, ISBN 0-262-63299-3.

A readable collection of papers on how people make use categories to solve problems quickly without a lot of effort: "Simple Heuristics That Make Us Smart" by Gerd Gigerenzer, Peter M. Todd and The ABC Research Group, published by Oxford University Press, ISBN 0-19-154381-7.

6.2 Acknowledgements

The authors wishes to thank everybody who volunteered their time to take part in the experiments and ACCU for making a slot available, in which to run the experiment, at both conferences.

Thanks to Matt Hearn, Brittany Babin and Oluwaseyi Fatadi of Loyola College for decrypting subjects' written answers into machine readable form.

Thanks to Michael Hahsler for suggestions on the use of his seriation package.

References

1. R. W. Bowdidge. Refectoring gcc using structure field access traces and concept analysis. In *Procedings of the Third International Workshop on Dynamic Analysis (WODA 2005)*, pages 1–7, May 2005.
2. T. Brants and A. Franz. *Web 1T 5-gram*. Linguistic Data Consortium, Philadelphia, USA, 1 edition, 2006.
3. K. Cwalina and B. Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries*. Addison–Wesley, 2006.
4. S. A. Gelman and E. M. Markman. Categories and induction in young children. *Cognition*, 23:183–209, 1986.
5. M. Hahsler, K. Hornik, and C. Buchta. Getting things in order: An introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, Mar. 2008.
6. E. Heit. Background knowledge and models of categorization. In U. Hahn and M. Ramscar, editors, *Similarity and Categorization*, chapter 9, pages 155–178. Oxford University Press, Apr. 2001.
7. D. M. Jones. I_mean_something_to_somebody. *C Vu*, 15(6):17–19, Dec. 2003.
8. D. M. Jones. Experimental data and scripts for short sequence of assignment statements study. http://www.knosof.co.uk/cbook/accu04.html, 2004.
9. D. M. Jones. Experimental data and scripts for developer beliefs about binary operator precedence. http://www.knosof.co.uk/cbook/accu06.html, 2006.
10. R. Krovetz. Viewing morphology as an inference process. Technical Report UM-CS-1993-036, University of Mass-Amherst, Apr. 1993.
11. W. T. Maddox and C. J. Bohil. Costs and benefits in perceptual categorization. *Memory & Cognition*, 28:597–615, 2000.
12. B. C. Malt, S. A. Sloman, S. Gennari, M. Shi, and Y. Wang. Knowing versus naming: Similarity and the linguistic categorization of artifacts. *Journal of Memory and Language*, 40:230–262, 1999.
13. B. C. Malt, S. A. Sloman, and S. P. Gennari. Universality and language specificity in object naming. *Journal of Memory and Language*, 49(1):20–42, 2003.
14. R. E. Nisbett and A. Norenzayan. Culture and cognition. In D. Medin and H. Pashler, editors, *Stevens' Handbook of Experimental Psychology, Volume Two: Memory and Cognitive Processes*, chapter 13. John Wiley & Sons, third edition, Apr. 2002.
15. E. M. Pothos and N. Chater. Rational categories. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, pages 848–853, 1998.
16. B. H. Ross and G. L. Murphy. Food for thought: Cross-classification and category organization in a complex real-world domain. *Cognitive Psychology*, 38:495–552, 1999.