Analysis of Design Processes by Function, Behavior and Structure

— Preliminary Reports —

Hideaki Takeda
Nara Institute of Science and Technology*
Masaharu Yoshioka and Tetsuo Tomiyama
The University of Tokyo†
Yoshiki Shimomura
Mita Industrial Co., Ltd.‡

Abstract

Function is a key concept to integrate object modeling and process modeling in design. In this paper, function is defined as a part of FBS (Function-Behavior-Structure) diagram, where function is a description of behavior abstracted through recognition of behavior for utilization. Function defined above is then used in FEP (Functional Evolution Process) to represent design processes. In FEP, function can be evolved in four ways, i.e., *decomposing evolution, causal evolution, "patch" evolution* and *modificatory evolution*. We analyze the design process of the team design by FBS and FEP scheme. We extracted 37 different function states and 77 different structure states. Firstly we compare the function model of assignment, users' trial and evaluation, and the designed object at the final stage so that evolution of requirements are explicitly represented. Secondly we examines each evolutional step of design. We extract 110 steps of changing of function and structure states and classify into three types of function evolution process, four types of functional modifier evolution process, and one structure evolution process.

Introduction

Function is a key concept in design because ideally design is a process in which object is realized from its functionality (see [1]). Although function is well known concept, its definition has been vague yet. In our approach, function is defined using structure and behavior (FBS: Function-Behavior-Structure modeling)[2]. Then function is used to evolve design so that function is gradually evolved (FEP: Functional Evolution Process).

There are three roles for function in design. Function is used firstly as a modeling language by which designers can compose and develop their requirements. It also serves as object representation which can connect requirements and objects. After construction and deliberation of objects function representation is used to evaluate objects to know how much their intention is satisfied.

*8916-5, Takayama, Ikoma, Nara 630-01, Japan, Fax: +81-7437-2-5269, Email: takeda@is.aist-nara.ac.jp
†Yayoi 2-11-16, Bunkyo-ku, Tokyo 113, Japan, Fax: +81-3-3813-5772, {yoshioka, tomiyama}@zzz.pe.u-tokyo.ac.jp
‡Tamatsukuri 1-2-28, Chuo-ku, Osaka 540, Japan, Fax: +81-6-764-3309, simomura@mita.co.jp

In the following sections, we will show our model of function and a test case of its application. We will explain FBS modeling in Section 2 and FEP modeling in Section 3. Then we will show results of the analysis of design process of the team design by FBS/FEP modeling.

1 Function-Behavior-Structure Modeling

There are many approaches to represent function, but there is a common problem, i.e., function and behavior are confused and mixed. Behavior can be directly derived from structure of objects and their environment, while function is related to not only structure of objects and their environment but also related to perception of object by designers. For example, suppose function of a car. Some people may say one of its function is "moving", others "carrying", and others "trampling", even if they observe the same behavior. Therefore we distinguish function, behavior, and structure levels in object representation (see Figure 1).

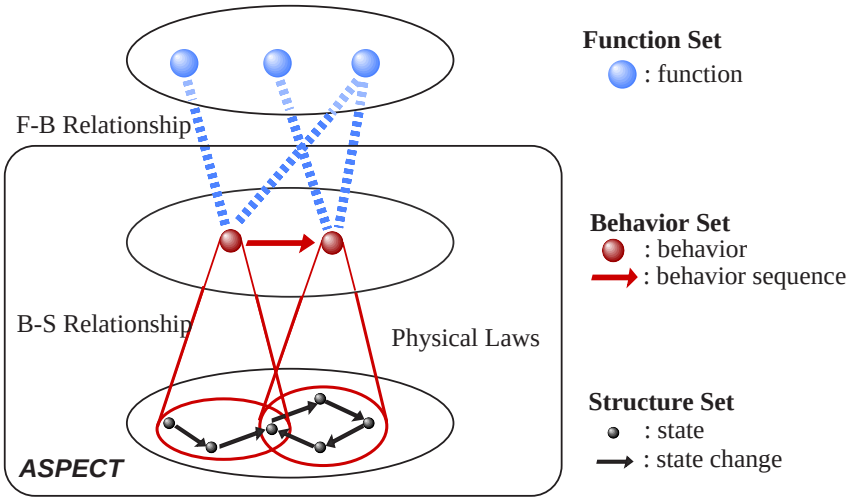

Figure 1: Relationship among Function, Behavior, and Structure

Structure level is represented by entities, attributes of entities, and relations among entities. Entities are identifiers of objects, and attributes of the entities and relations among the entities represent structures composed by the objects. Then behavior is defined as "sequential change of states of objects." In the physical world, changes of states of objects are governed by physical laws. We call this set of definitions of structure and behavior *aspect* which is a basic unit of object representation. Aspect consists of definition of terms and entities (structure) and rules (physical laws). Designers have many kinds of aspects from well-defined aspects (e.g., rigid body dynamics) to ill- or vaguely- defined aspects (e.g., manufacturability).

While behavior is grounded on structure and within the scope of aspect, function is indirectly related to structure and not in the scope of aspect. We define function "a description of behavior abstracted through recognition of behavior for utilization." Function is defined on a chunk of behavior (or behavior itself). There are a lot of possible chunk of behavior. But only some of them are meaningful for designers when they recognize and design objects.

Although function is not included in aspect, most of functions are associated to aspects,

because behaviors which a function is based on, are in a single aspect. In other words, an aspect has a set of associated functions which is a description of the aspect in the view point of utilization.

A function is represented as combination of a *function body, objective entities,* and *functional modifiers.* A function body is a symbol which carries meaning of the function. A typical function body is a verb word in sentences like "move" and "carry." An objective entity is an entity which the function occurs on or to. It should be realized as an object in structure level until the end of design. A functional modifier is a symbol which restricts function in order to match the functionality with designers' intention. A typical functional modifier is an adverb word like "precisely" or "firmly."

The difference between function body and functional modifier is degree of satisfaction (see Figure 2). Satisfaction of function is usually 'yes' or 'no', that is, we recognize whether a function exists on an object or not. There are no intermediate states. On the other hand, functional modifier has degree of satisfaction. We can judge how much "attach firmly" is achieved on an object. We can also compare two objects by degree of satisfaction of functional modifiers. In other words, functional modifiers are indexes to characterize how a function is achieved. A function has indexes as many as its functional modifiers.

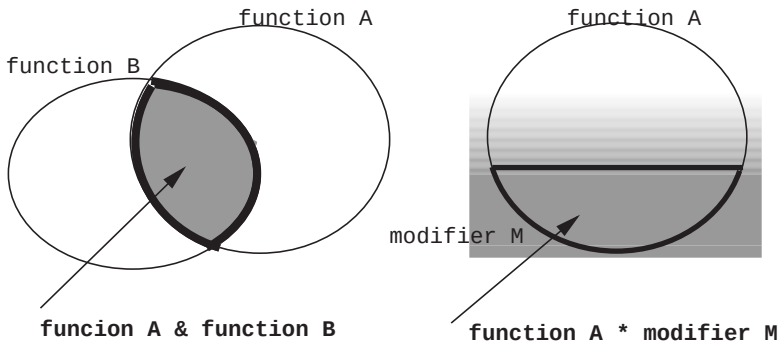

Figure 2: Difference between function and modifier

2 Functional Evolution Process

In this section, we show function representation in design processes.

One of the three roles of function is a language to describe requirements. Requirements are not complete at the beginning of design in almost every design process. Details of requirements are realized according to detailing of object descriptions, i.e., function is also detailed in design processes. We call this detailing process of function *functional evolution process.*

In order to represent functional evolution process, we provide description of function and its relationship.

We provide three types of relations among functions.

decompose It is a typical process for designers to divide a function into sub-functions. This relation should be transferred to behavior and structure level.

be-caused-by It means that new function B is needed to exist in order to realize function A. In other words, B is necessary condition for A. This relation should be supported causal relation in behavior level.

be-reinforced-with It means that new function B is recommended to exist in order to realize function A properly. In this case, since B is not necessary condition for A, A can exist without B. But A with B would accomplish its functionality more properly. This relation would be generated as a result of interpretation of functional modifiers.

Functional evolution is to generate functions and relations among functions. According to these three types of functional relations, functional evolution has three different ways.

decomposing evolution Designers try to find sub-functions from a function. Then they try to find either sub-sub-functions or behaviors associated to sub-functions (see Figure 3). For example, Function "to visualize weight" is decomposed into Function "to make weight into displacement" and Function "to convert weight and visualize".

causal evolution Designers try to find functions linked by causal relation. This relation is found through behavior level. First they would find behavior associated to the given function. Then they would find causal behaviors to the behavior by using causal simulation (e.g., Qualitative Simulation[3]). Finally they would obtain functions associated to these behaviors (see Figure 4). For example, Function "to translate weight into displacement" invokes Structure "spring". But by mental simulation designers find that a new structure like "plate spring" is needed "to guide" spring. Function "to guide" is found through behavior and structure levels.

"patch" evolution Designers would find a new function by consulting functional modifiers. Then designers would combine and test behaviors associated to the given function and the new function in order to know whether the new function would support the given function according to the modifier (see Figure 5). For example, Function "to enumerate rotation" invokes Structure "rotation plate". Then designers examine how rotation plate can realize Function "to enumerate rotation" with modifier "as large as possible", here the modifier is criterion to evaluate realized function. Then designers find another function "to enlarge indicator" is needed to accomplish the function properly. This function can not be derived in behavior and structure levels only, but functional evaluation can generate it.

modificatory evolution While the above three types of evolutions are evolution of function, modifiers are also evolved in design processes. Designers try to add new functional modifiers to existing functions. The important motivation to add new modifiers is evaluation of functionality of objects (see Figure 6). By comparing the desirable functionality and functionality of design object, designers would add new modifiers to shorten the gap.

Through these processes, functional representation is gradually detailed.

3 Analysis of the protocol data

3.1 Modeling Method

We used the protocol data of the team design up to time 1:49. The reason to choose the team design is that it is relatively easy to understand because verbalization is clearer. We believe that protocol analysis on design should be done with more than two designers (see Ref. [4]).

We extract FBS elements in design processes, i.e., functions, functional modifiers, behaviors, and structures. Extraction here is not pure grammatical operation but heuristic operation which is dependent to contexts. We use grammatical information as hints to find these types of information (for example, verb words for functions).

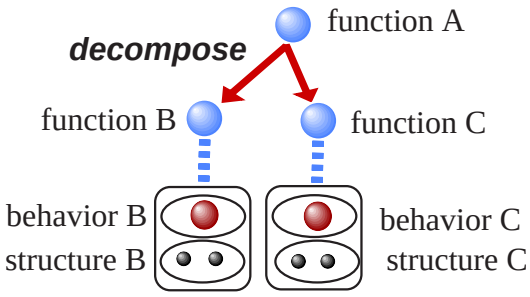

Figure 3: Decomposing Evolution

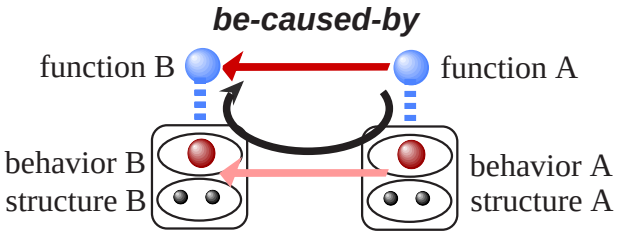

Figure 4: Causal Evolution

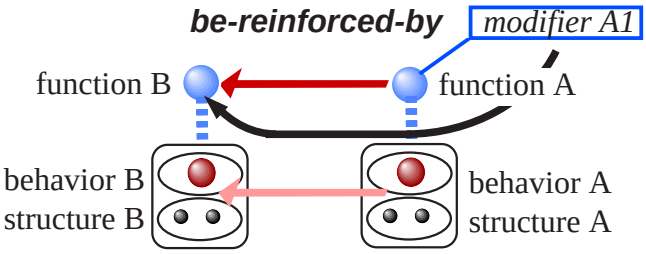

Figure 5: "Patch" Evolution

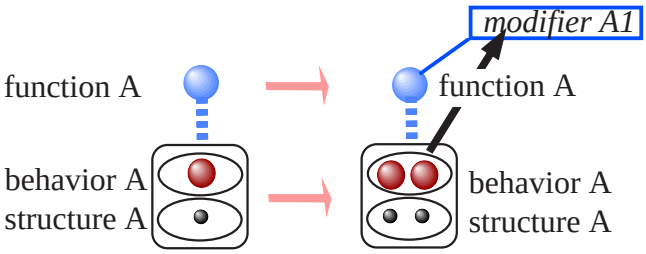

Figure 6: Modificatory Evolution

In this paper we use lisp-like forms *(function-body subjective objective1 ...)* to represent functions just for convenience. For example, a function description "A device can *carry/fasten* a backpack to a bike" is represented *(carry/fasten device backpack bike)*.

Criterion to extract these types of information is as follows;

function We regard verbs to explain objects as functional description. In addition, some special words like *feature* also indicate functions of objects. We pick up a verb with a subjective word and objective words as a function.

functional modifier We interpret adverbial phrases to verbs which stand for function as functional modifiers. Furthermore we also interpret additional conditions or additional explanations to function.

behavior Behavior is appeared when designers invoke simulations. Problem is that such simulation is often done by non-verbal actions. For example, designers operate physical objects to simulate some motion. They also simulate motions on papers. We observe designers' action to detect their simulation.

structure Since structures are what is designed, they appeared as noun phases mainly. But part of some structures are also non-verbal because they are found in figures. We gather verbal and non-verbal information to describe structures. As we restrict ourselves in symbolic representation in this analysis, we represent a structure as a set of concepts, relations among concepts, and attribute of concepts.

After extraction of information on FBS elements, we construct descriptions of the FBS model at each step of the design process, which we call FBS *state* model. A FBS state model should consist of three elements of FBS, that is, functional, behavior, and structure. Since we focus our attention on transition of function and structure, we made state models in the following two ways.

Firstly we model functions and functional modifiers at each step of the design process. Each time functions or functional modifiers are added newly or changed, we gather all functions and functional modifiers which are under consideration up to that time. A *function state model* or *function model* consists of these functions and functional modifiers. Function models are linearly ordered according to time stamps.

Secondly we focus our attention on structures and behaviors that the designers propose in design. Each time structures are added newly or changed, we gather descriptions of the structure which are under consideration up to that time. If behaviors of the structures are considered, we also gather descriptions of their behaviors. A *structure state model* or *structure model* consists of these structures, behaviors, functions and functional modifiers. Since a new structure model may be based on earlier structure models, Structure models can be partially ordered according to this relation.

We can get a FBS state model at certain time by combining a function model and a structure model which is used at that time (see Figure 7).

Figure 8 show examples of relations among function and structure state models. A symbol started from `f` indicates a function model, a symbol started from`s` indicates a structure model. Following words to a structure model is its short explanation. A allow line indicates relation between structure models.

We made 37 function models and 77 structure models.

3.2 Specification Evolution

In this design process, there are mainly three types of information for objects. One is the assignment which is specifications each of which the product *should* satisfy in their design. Second one is the market research and users' trials and evaluation of the prototype. It is

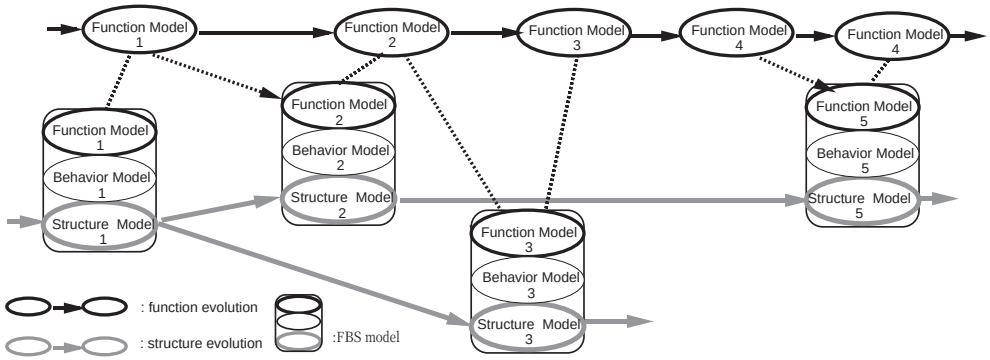

Figure 7: Combination of function and structure models

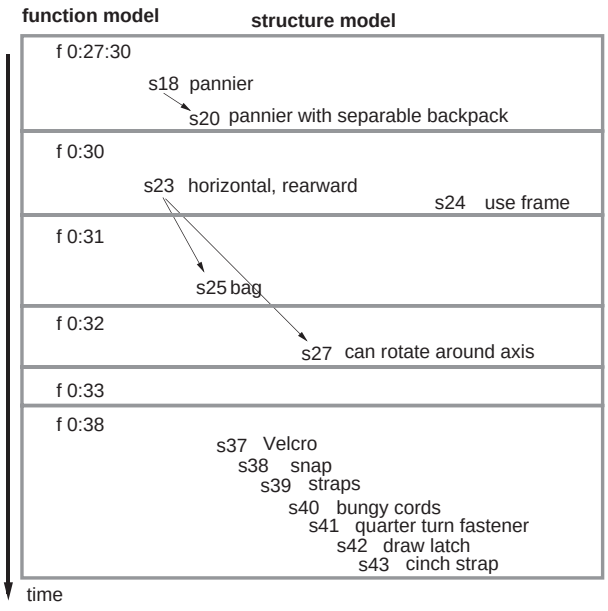

Figure 8: Function and structure state models

what the product had better satisfy but is not necessary. These two types of information are directly related to the product itself. Third one is details of environments of the objects, i.e., details of the bike and the backpack which define the environment where the product is used.

Figure 9 shows the FBS model of the assignment. In this figure, we identify three independent functions, i.e.,

1. `(carry/fasten device backpack bike)`
2. `(stack-away device)`
3. `(fold-down device)`

A thin black line indicates relationship between a function body and a objective or subjective word. And function `(carry/fasten device backpack bike)` is decomposed into these two functions[1].

1. `(attach device backpack)`
2. `(attach device bike)`

A thick black arrow indicates decomposition of a function. There are four functional modifiers in the FBS model of Assignment, i.e.,

1. `(easy-of-use (carry/fasten device backpack bike)`
2. `(a-sporty-appealing-form (carry/fasten device backpack bike)`
3. `(for-most-bikes (carry/fasten device backpack bike)`
4. `(reasonable-price-range (carry/fasten device backpack bike)`
5. `(easily (stack-away device))`
6. `(easily (fold-down device))`

Both of modifiers `easily` are decomposition of modifier `ease-of-use`.

In this figure, a box indicates a modifier, a grey line indicates relationship between a modifier and a function body, and a grey arrow indicates decomposition of a modifier.

This representation is good enough to understand what is the required functions of this device. However it is too vague to design because this representation covers a huge number of candidates, i.e., a huge design space. Therefore designers need to reduce the possible design space.

To reduce the possible design space in this design session, they can consult users' trial and evaluation on prototype design. Figure 10 shows the FBS model of the assignment and users' trials and evaluation.

Functions appeared in this figure are the same with those in Figure 10, but fourteen modifiers are added to FBS model of Assignment. As we discussed in the previous section, adding modifiers is to set index to evaluate how functions are achieved. So these newly added modifiers are good information for designers to realize how the design space should be reduced.

Since these modifiers are not necessary to satisfy, designers picked up some of them to consider in design session and they left others untouched. `?` marked modifiers are not appeared in the design session.

Figure 11 shows the final FBS model in our analysis of the design session.

9 new functions are added to initial FBS model, i.e.,

[1] In Assignment a word `fasten` is used instead of `attach`. But we choose `attach` rather than `fasten` because it is used more commonly later in the design session.

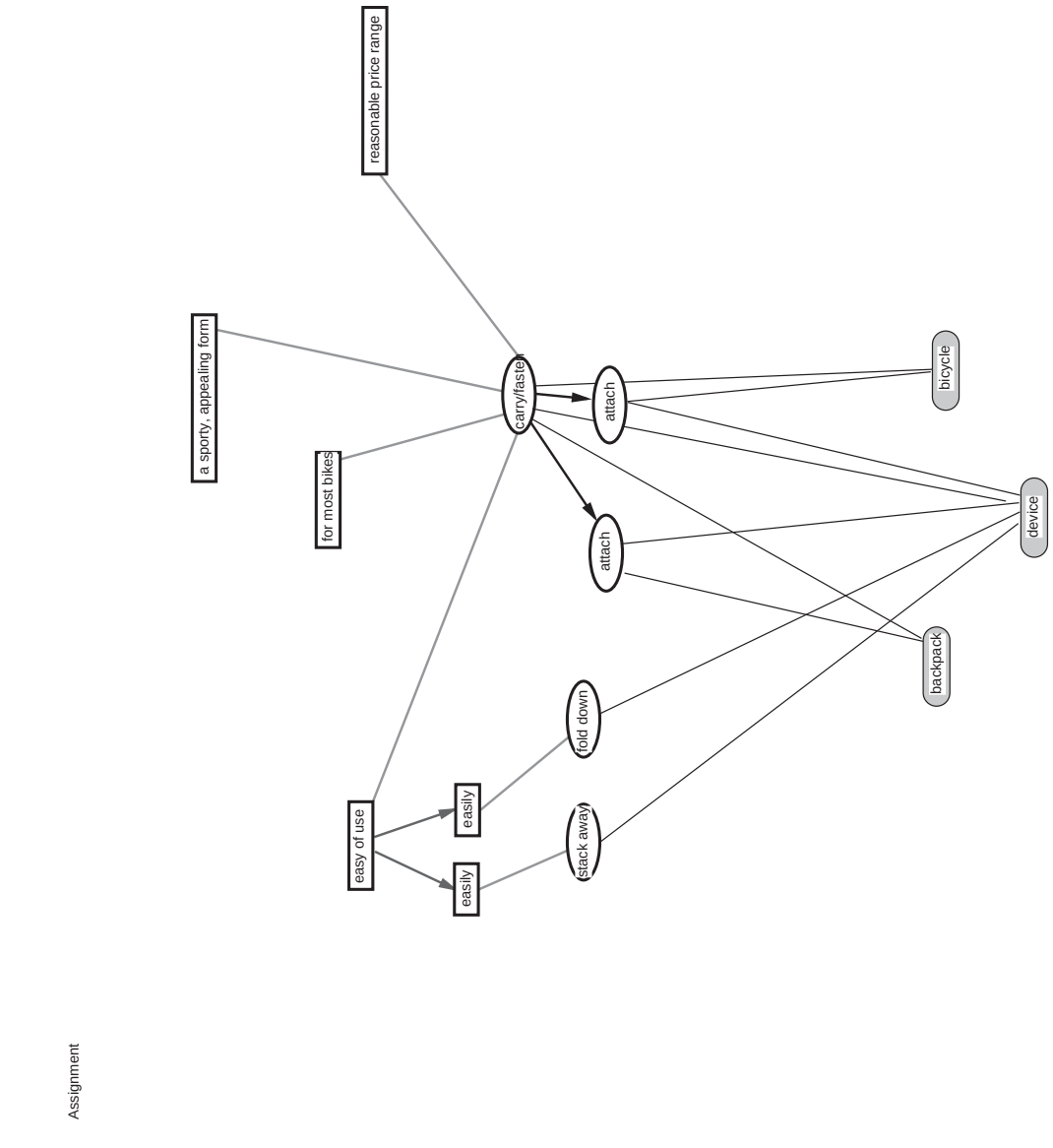

Figure 9: Function model of the assignment

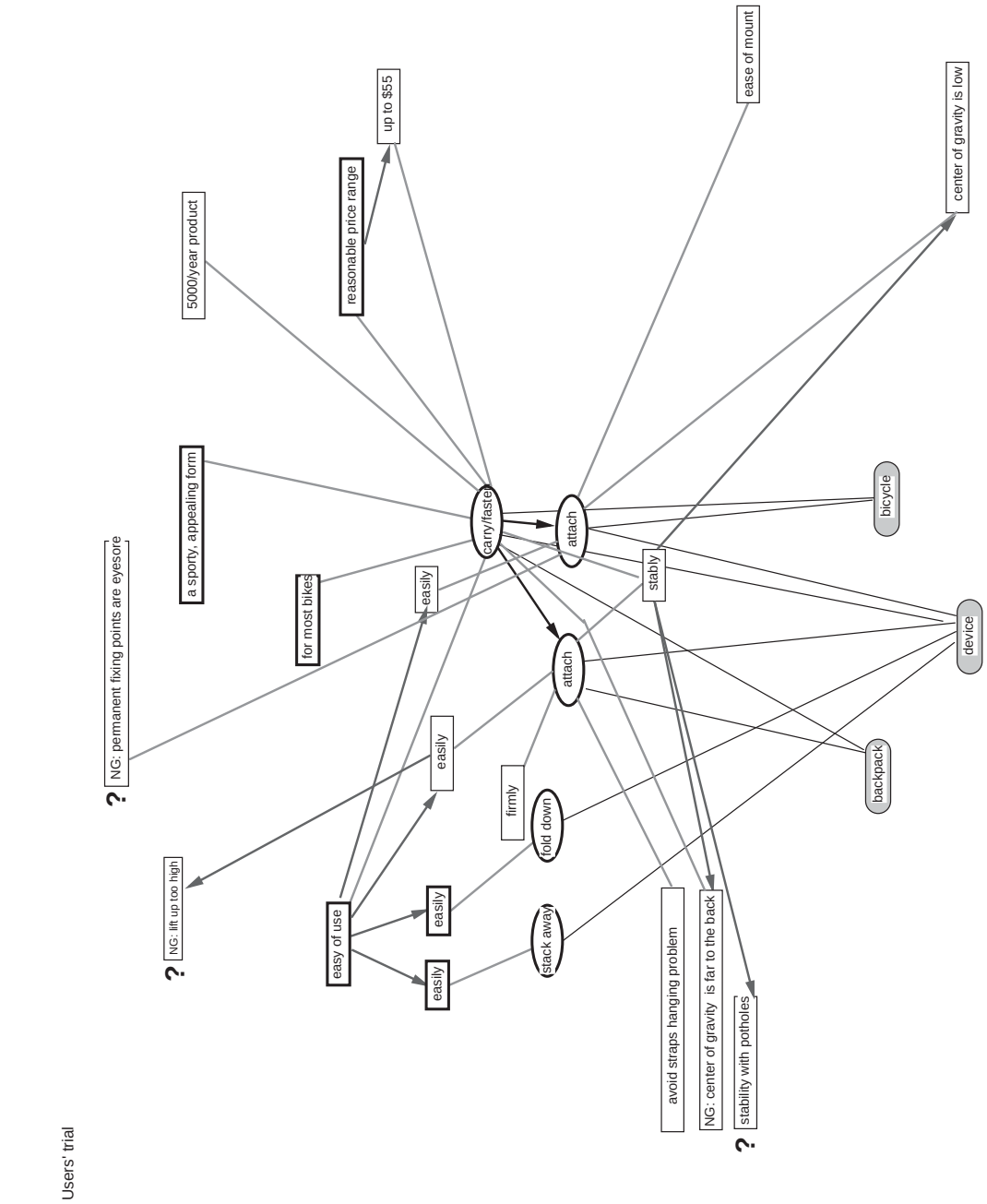

Figure 10: Function model of the assignment and users' trials and evaluation

1. `(move person bicycle)`
2. `(steer person bicycle)`
3. `(pedal person bicycle)`
4. `(support device backpack bicycle)`
5. `(put-things-in device)`
6. `(stand-up device bicycle)`
7. `(stand-up device)`
8. `(rack-function device)`
9. `(fender-function device)`

It does not means that all those functions are necessary to satisfy, but they are potential functions that the device may have. For example, the design solution at 1:49 will not satisfy 5th, 6th, 7th, and 8th functions. But "tray, net, and drawstring" design, an abandon design solution would satisfy 5th function.

31 modifiers are added to the initial FBS model. Since 9 modifiers are appeared in users' trial FBS model in these modifiers, 22 modifiers are newly found in the design session.

These newly added modifiers are not necessary to satisfy in the same way as newly added functions.

We focus our attention on function `(attach device bike)` here to see how functional representation is changed in the design process.

There are no modifiers for `(attach device bike)` in Assignment FBS model. Modifiers for `(attach device bike)` in Users' trial FBS model are

1. `easily`
2. `center-of-gravity-is-low`
3. `ease-of-mount`
4. `permanent-fixing-point-is-eyesore`.

Modifiers for `(attach device bike)` in the final FBS model are as follows;

1. `easily`
2. `center-of-gravity-is-low`
3. `ease-of-mount`
4. `good-side-balance`
5. `adjustable-for-most-bikes`
6. `not-heavy-to-steer`
7. `not-too-low-to-fit-things`
8. `not-to-difficult-to-pedal`
9. `reduce-the-number-of-parts`

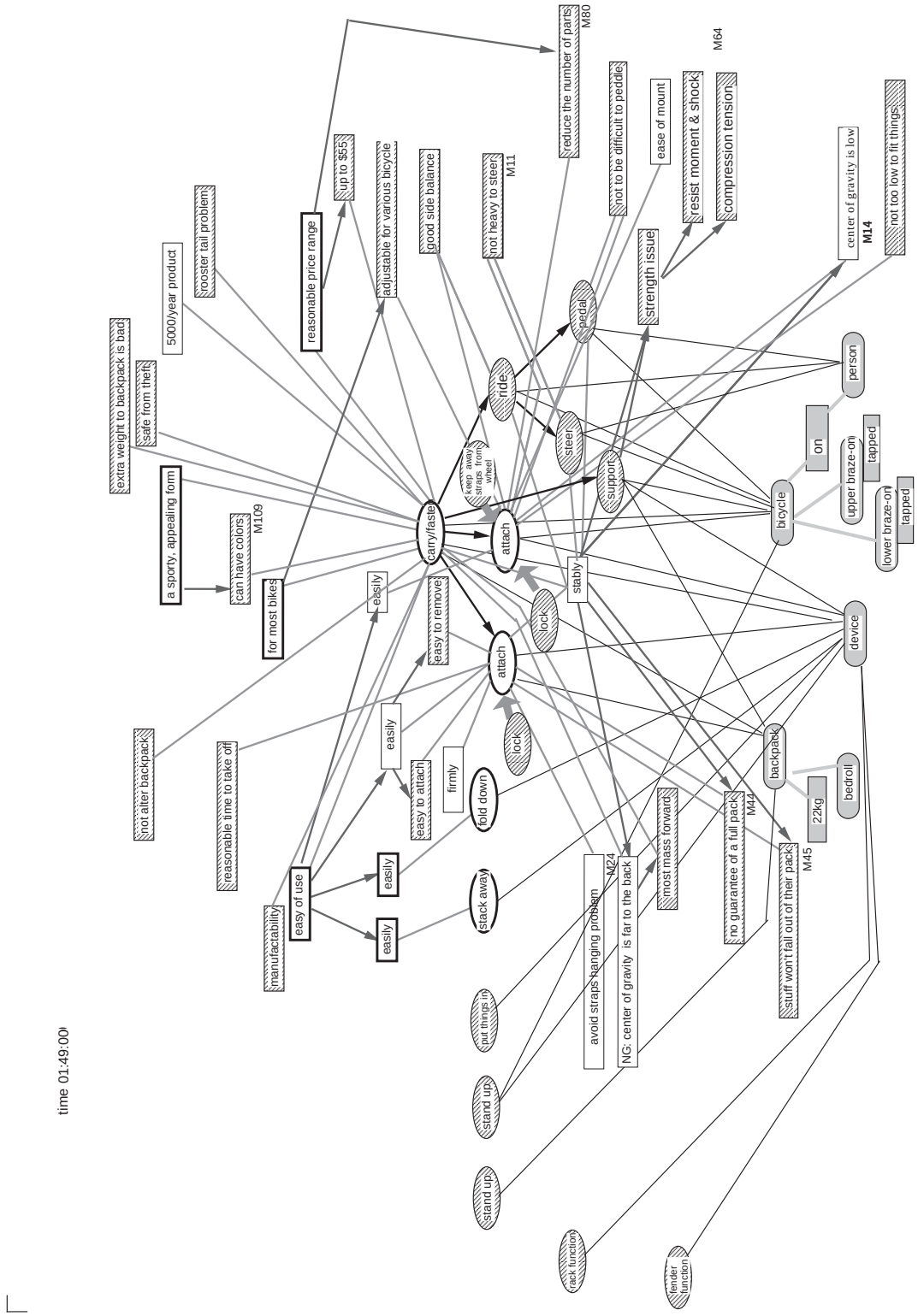

Figure 11: Function model at 1:49

Type of evolution	Number
Decomposing functions	2
Adding functions by structures	5
Adding functions by additional behavior	1
Adding functions by transforming modifiers	3
Adding modifiers negatively by structures	10
Adding modifiers positively by structures	8
Adding modifiers by other reasons	10
Adding structures	71
Total	110

Table 1: Numbers of appearance of evolutional steps

First three modifiers are inherited from users' trial FBS model, and other six modifiers are found newly.

Some of them are come from other modifiers. `(adjustable-for-most-bikes (attach device bike))` is modifier decomposition from `(for-most-bikes (carry/fasten device backpack bike))` according to function decomposition. `(reduce-the-number-of-parts (attach device bike))` is modifier decomposition from `(reasonable-price-range (carry/fasten device backpack bike))` according to function decomposition. Others are found in constructing and evaluating new structures.

It means that this `attach` function is described in more detailed way.

3.3 Analysis of evolution steps

In this section, we investigate how each function or functional modifier is developed in the design process.

Table 1 shows nubmbers of appearance of each evolution types.

3.3.1 Adding functions

A new function is added by the following three ways.

Decomposing functions It is claimed that decomposition of functions is the most basic procedure to handle functions in design (for example see Ref.[5]). Unless the designing domain is well investigated or well known like routine design, it is not easy to find decomposition of functions. Actually in this design, decomposition of function is found only twice.

Designers suggested attaching the device to handle bar of the bicycle around time 0:27. Then they found that it made heavy a rider to steer with the backpack. Then they decided to suspend developing this structure.

At this moment, they found a new function `steer` which is a sub-function of `ride`, and also found a new modifier `heavy to steer` to function `steer`.

Adding functions by structures A new function is added by examining a newly suggested structure model. It is similar to decomposing functions, but added functions are auxiliary or unexpected functions so that they are not relevant to existing functions.

At time 1:07, one of designers suggested a propylene popup-book-like design. The advantage of this design he claimed is that it would work as mudguard (see Figure 12).

Although he retracted this idea immediately because of strength problem, this mudguard function is used again later as fender function when they discussed function of the vacuumed formed tray (at time 1:22).

Adding functions by additional behavior This is the procedure to make causal relations between functions.

At time 1:06, they discussed the strength issue to attach the device to the bicycle. They simulate dynamics of the bicycle with the device and the backpack in order to know how strength of parts would affect the behavior. Using this simulation, they realized function `(support device backpack)` is needed as a sub-function of function `carry/fasten` because both behavior `support` is needed so that behavior `attach` is realized[2].

Since most of behavior are implicit or non-verbal, it is difficult to find such causal relations.

Adding functions by transforming modifiers It is "patch evolution" of function we explained in the previous section. Although we observed many modifiers in this design, it is seldom to use this procedure, mainly because of simple functionality of the designing object. Since functions given as requirements and expected structures are simple, we do not expect many sub-functions to realize the given functions.

In time 01:30, the designers try to satisfy modifier `safe-from-theft` (this modifier is already suggested in time 00:38).

One idea to satisfy it is to understand `(safe-from-theft (attach device backpack)` as `(safe-from-theft (fasten/carry device backpack)`. Then they suggested that function `lock` performed by `lockable knob` is realization of this modifier. This function `lock` is a reinforcing function to function `attach` (see Figure 13) .

3.3.2 Adding modifiers

As we mentioned, there are a few functions and many modifiers in this design. It means that adding modifiers is important process to evolute function models.

A typical procedure observed in the design process is adding modifiers by examining new suggested structures.

Figure 14 shows how addition of modifiers is achieved. Firstly a function model and a structure model are taken (Figure 14(a)). Comparing the function model and the structure model, designers suggest new structure model (Figure 14(b)). Behaviors are produced by simulation on this new structure model, and then function model on this structure model is created (Figure 14(c)). Actually this new function model is represented as difference from the previous function model. If the new function model is considered better than the previous, modifiers representing the difference are to be included descriptions of functionality. If the new function model is considered worse, negation of modifiers representing the difference are to be included (Figure 14(d)).

Adding modifiers negatively by structures Around time 28:00, they suggested to use panniers to attach the device to the bicycle (see Figure 15). Then they found that to attach the backpack to a single side of panniers would be bad to keep side balance. They decided that the pannier structure was not desirable because of this reason.

In order to find the balancing problem, they might simulate the behavior of the device with some aspect. The behavior aspect of this state is dynamics on driving a bicycle. Simulation of dynamics on driving a bicycle with the device and the backpack should tell

[2] Concept `support` and concept `attach` are used either as function or behavior.

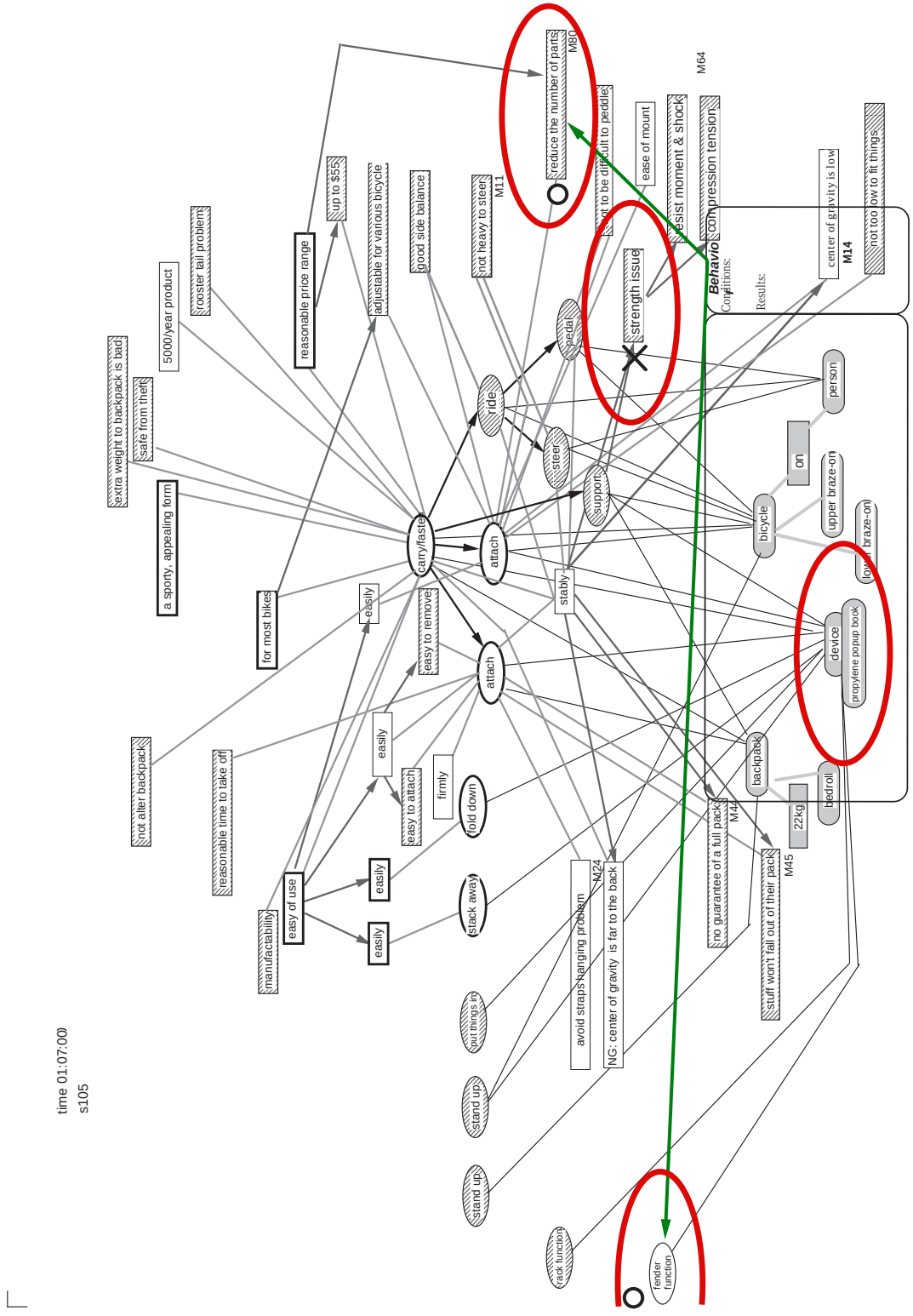

Figure 12: An example of adding functions by structures

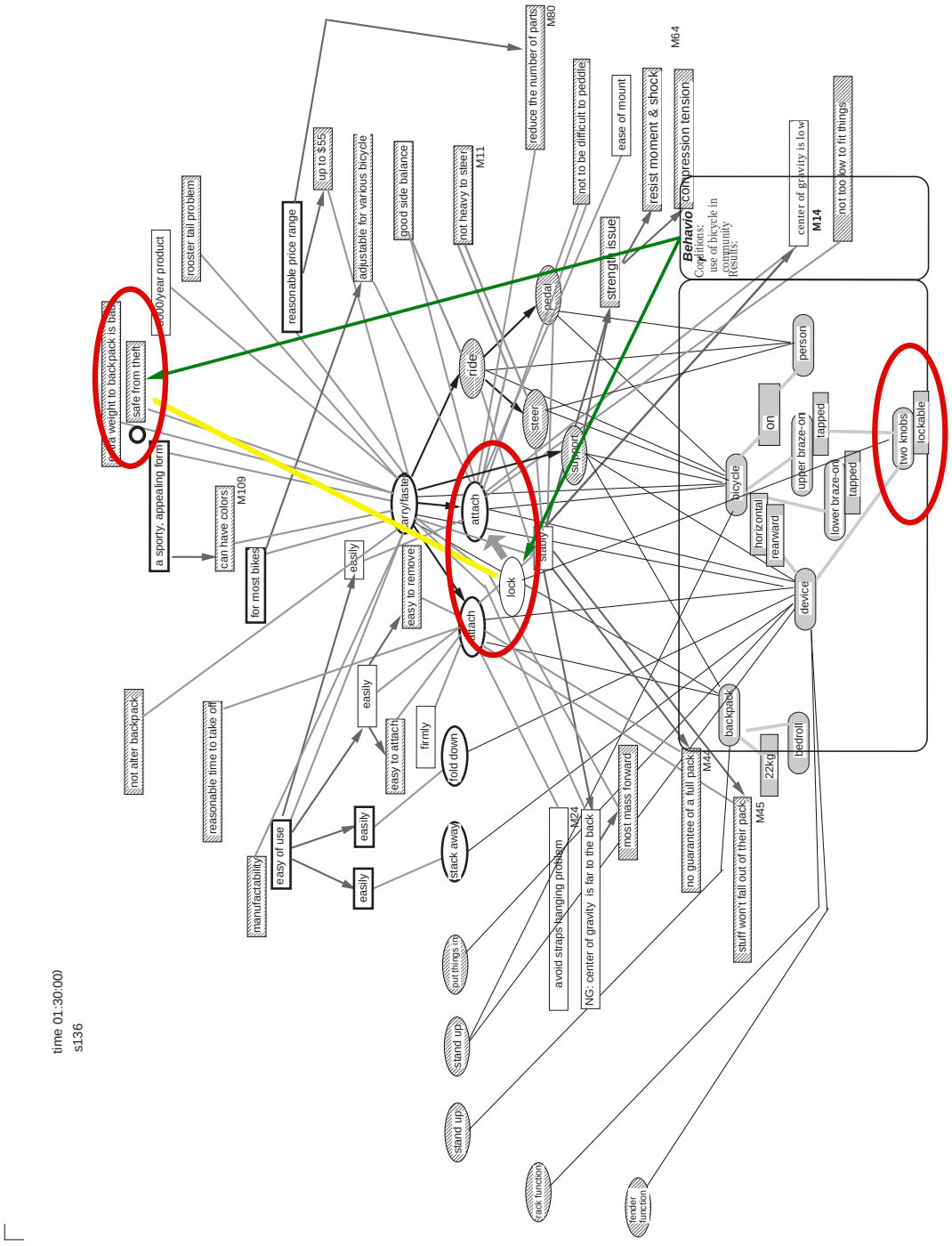

Figure 13: An example of Adding functions by transforming modifiers

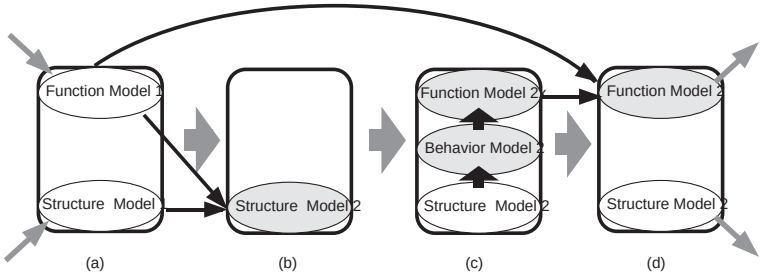

Figure 14: Adding modifiers by structures

them that weights of both sides would be not in balance. Then designers interpreted that it is bad.

This decision means not only decision on the structure, but also decision on the functional model. At this moment, they found that riding should be with good side balance, i.e., they found a new specification to function *ride*. Thus *good-side-balance* is a new modifier to function *ride*, and is added to the functional model.

The same procedure happened just after this state. They proposed a new structure, i.e., separation of the backpack, which is to satisfy *good-side-balance* modifier. Then they noticed that altering the backpack is not desirable. Thus *not-alter-backpack* is also added to the function model.

Adding modifiers positively by structures At time 01:15, they suggested to use aluminum as material of the device. They claimed that its advantage is to have colors.

Since they thought that *to-have-colors* is an idea to realize *a-sporty-appealing-form*, *to-have-colors* is decomposition of modifier *a-sporty-appealing-form*, and can be added to the functional model as a modifier to function "carry/fasten".

It is interesting that adding modifiers by a negative example is appeared more frequently than adding modifiers by a positive example. It suggests that contradiction or inconsistency is important to proceed design[3].

4 Summary

In this report, we showed our primary results of protocol analysis by our functional modeling scheme. Although it is tentative report, we can draw some remarks from these results.

- **Functional changing is important as well as structure changing to trace design processes**
  We have been absorbed in structure changing when we wanted to know what happened in design processes. Functional descriptions are also changing in design processes. Both changing are important, but it is impossible to model them separately. Our approach is suitable in this purpose because function and structure is represented in a single scheme.

- **Functional evolution is not magic but rational in most cases**
  Designers often make design criteria by themselves in order to converge their design processes. Designers seem to have such criteria *a priori*. But such criteria (functional

[3] We discussed and modeled use of inconsistency in design in Ref [6]

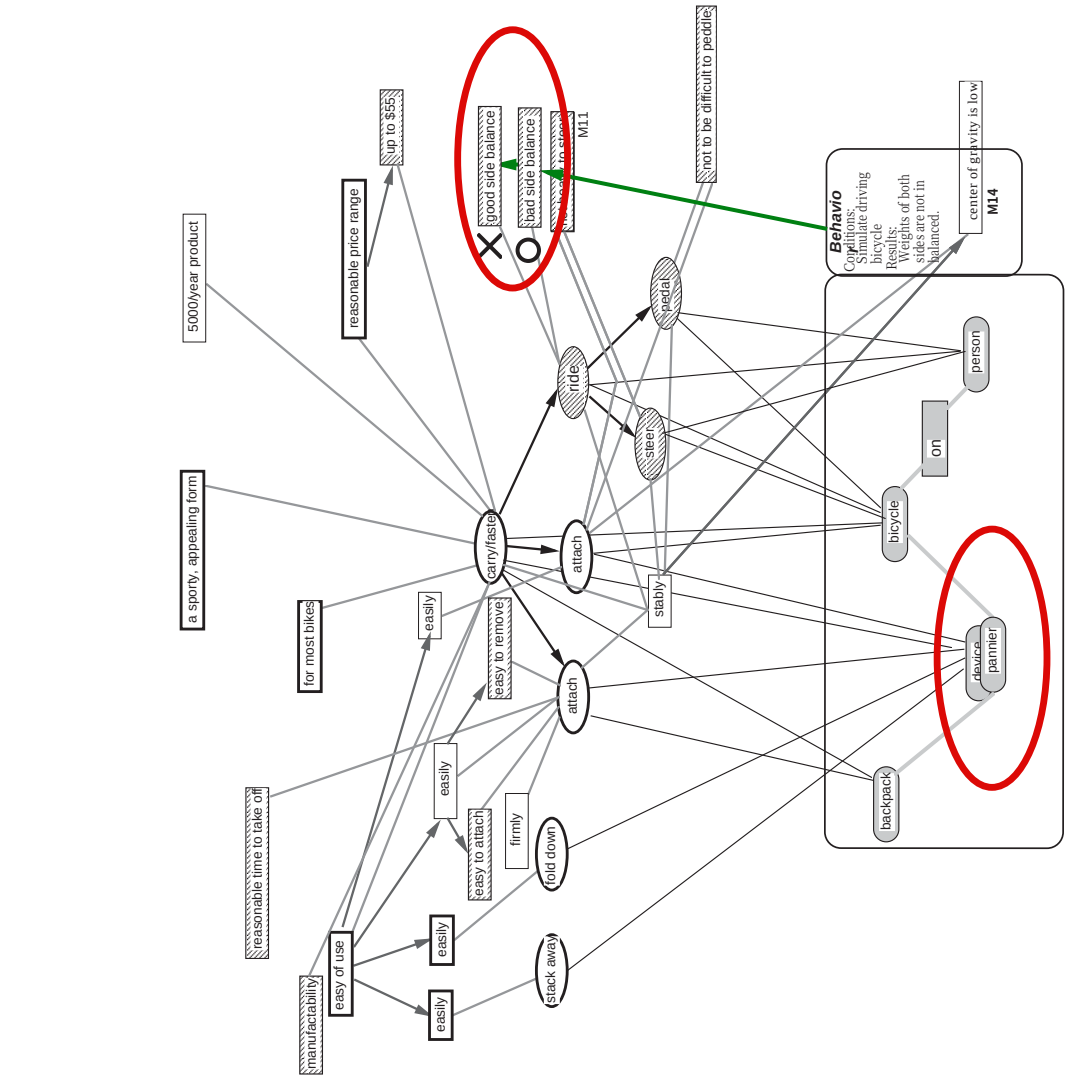

Figure 15: An example of adding modifiers negatively by structures

modifier in our term) is arisen as results of interaction between structure and function. We can explain why they adopt new criteria in our scheme.

- **function model is also important as result of design**
  Not only the structure they designed but also its functionality they intended are important as the result of design, because we can evaluate how much the designed structure would match the intention of designers. Functionality they intended is not the function given by requirements, but the function model they made during the design process.

References

[1] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa. Modeling design processes. *AI Magazine*, 11(4):37–48, 1990.

[2] Y. Umeda, H. Takeda, T. Tomiyama, and Y. Yoshikawa. Function, behaviour, and structure. In J.S. Gero, editor, *Applications of Artificial Intelligence in Engineering V*, volume 1, pages 177–194, Berlin, 1990. Springer-Verlag.

[3] K.D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[4] H. Takeda, S. Hamada, T. Tomiyama, and H. Yoshikawa. A cognitive approach of the analysis of design processes. In *Design Theory and Methodology – DTM '90 –*, pages 153–160. The American Society of Mechanical Engineers (ASME), 1990.

[5] G. Pahl and W. Beitz. *Engineering Design.* The Design Council, London, 1984.

[6] H. Takeda, T. Tomiyama, and H. Yoshikawa. A logical and computerable framework for reasoning in design. In D.L. Taylor and L.A. Stauffer, editors, *Design Theory and Methodology – DTM '92 –*, pages 167–174. The American Society of Mechanical Engineers (ASME), 1992.