Specifying Biological Systems as Reactive Systems: Some Observations

Amir Pnueli

New York University and Weizmann Institute of Sciences (Emeritus)

Towards System Biology, Grenoble, Oct. 2007

Based on Joint work with:

Hillel Kugler Microsoft, Cambridge
Jane Hubbard NYU
Michael Stern Yale

and helpful insights by

Naaman Kam

Reactive Systems

Programs whose role is to maintain an ongoing interaction with their environment, rather than produce a final result upon termination.

Examples: Air traffic control system, Programs controlling mechanical devices such as a train, a plane, or ongoing processes such as a nuclear reactor.

Such programs must be specified and verified in terms of their behaviors.

Milestone 1: Identification of reactive programs as a unique and challenging class which must be formalized in terms of the program's behaviors.

In Pictures

Computational Programs: Are run in order to produce a final result on termination. They Can be modeled as a black box.

and specified in terms of Input/Output relations.

Example:

The program which computes

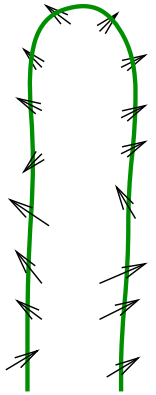
$$y = 1 + 3 + \cdots + (2x - 1)$$

Can be specified by the requirement

$$y = x^2.$$

On the Other Hand

Reactive Systems can be viewed as a green cactus (?)



Such programs must be specified and verified in terms of their behaviors.

Specification of Programs: Temporal Logic

A mathematical language for specifying behaviors. Reference to time through modal operators rather than explicit parameterization.

Note that, in spoken language, we also say

Tomorrow I will take a trip

rather than

At date = 11.10.07 I will take a trip

Main temporal operators are

- $\Box$ – Always
- $\Diamond$ – Eventually

Example: Specification of an Elevator

- It is never the case that the doors are open while the elevator is in motion.

  $\Box \neg(\mathit{doors\_open} \wedge \mathit{moving})$.

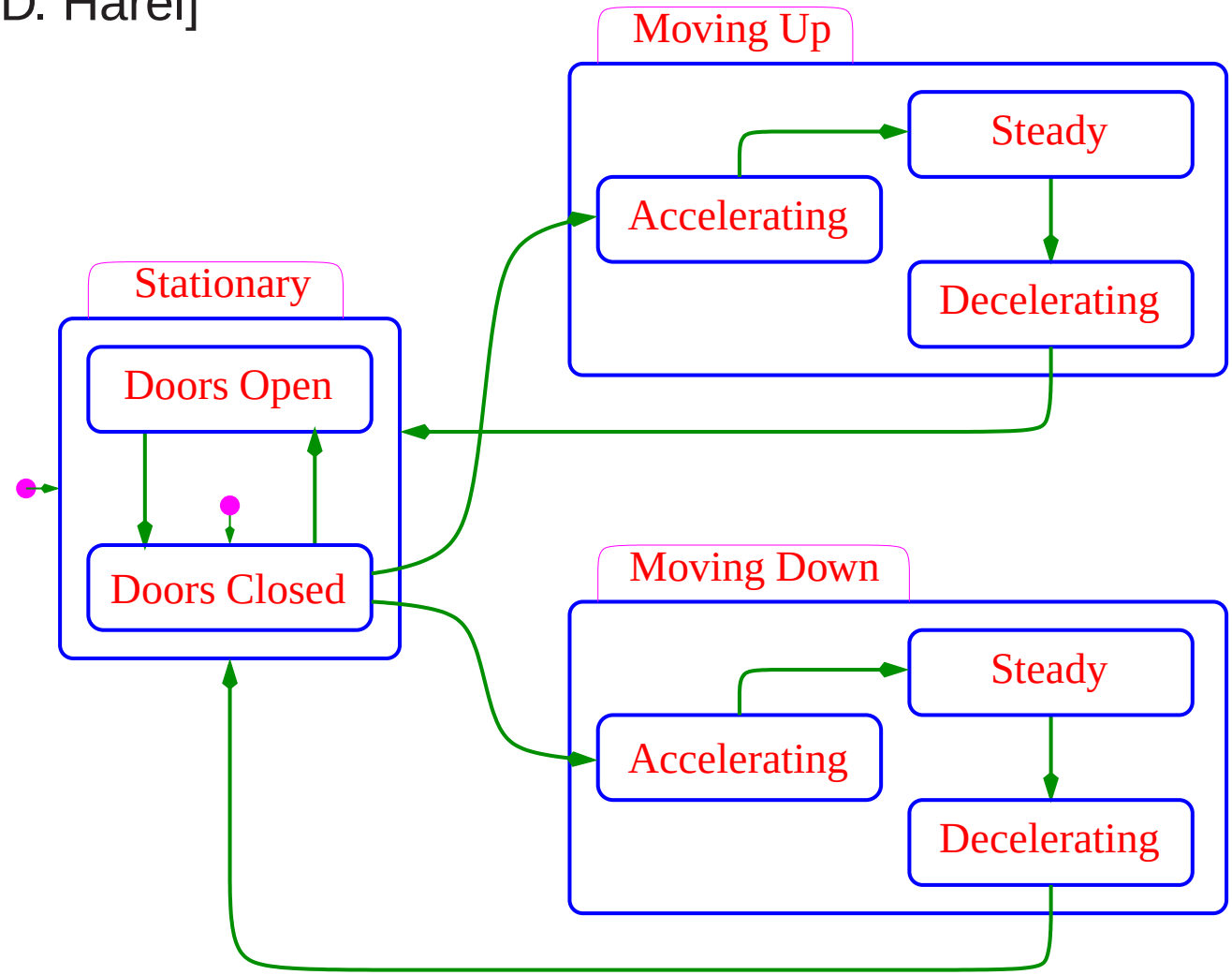
- Any request for floor 5 is eventually satisfied.

  $\Box (\mathit{request\_for}\,[5] \longrightarrow \Diamond\, (\mathit{at\_5} \wedge \mathit{doors\_open}))$.

Milestone 2: Develop Temporal Logic as a specification language for reactive systems.

A Visual Style of Specification

An alternate style of formal specification is provided by the visual formalism of Statecharts [D. Harel]



Application to Biological Modeling

Obviously, a biological system is one of the most quintessential reactive systems. Why not use the reactive-systems formalisms for its modeling?

A first successful attempt has been done by Naaman Kam who used the formal methods of Statecharts to model and verify the behavior of T-Cells in the immunological system. The mere need to construct a formal model as a reactive system gave rise to innumerable questions that were never asked before.

While modeling the system, it displayed a strange and unobserved phenomenon. The questions raised by this behavior led to investigations which yielded some new knowledge about additional actions which were being taken by the cell.

Moving on to C. Elegans

As the next biological system to be modeled as a reactive system, we chose C. elegans vulval precursor cell fate specification.

Unlike the previous example, here the interaction is not between the organism and its natural environment, but between the organism and the experimental scientist, who interferes with its natural development.

A further technical difference between the two models is that, in the C.Elegans model, we used the specification formalism of live sequence charts (LSC's) invented by Damm and Harel and its implementation by the play-in/play-out tool by Harel and Marelly.

Can Biology Benefit from Discrete Modeling?

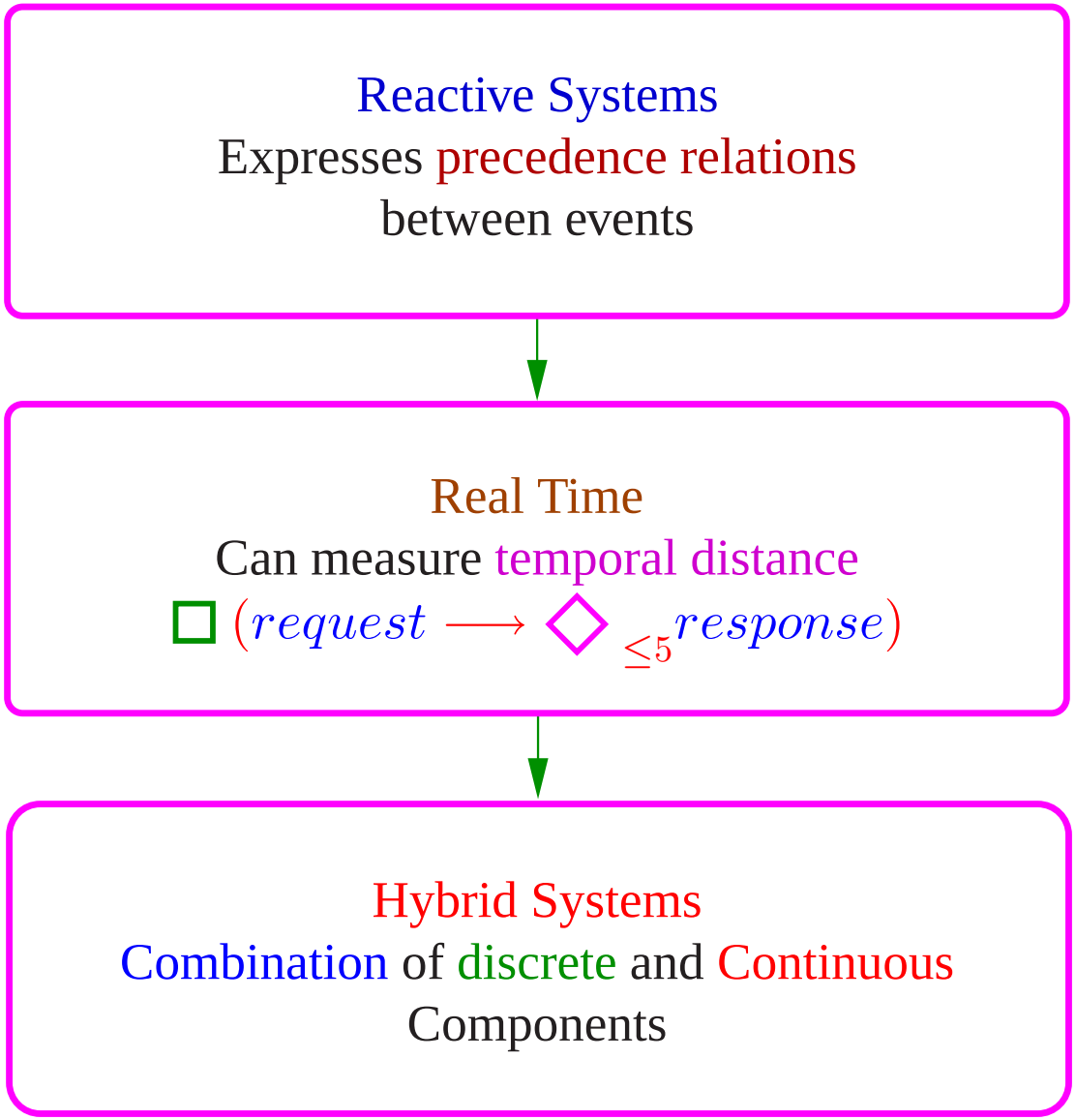
A basic question is whether the reactive-system modeling paradigm which is essentially discrete is appropriate/adequate for biological modeling.

Recall and compare some of the alternative suggested approaches:

- Differential equations. This is certainly the most precise model. However, we often do not know the values of the coefficients in the equations, and our computational ability is limited to the treatment of very small systems.
- Boolean equations. This can be viewed as a degenerate form of the reactive-system approach. Variable ranges are limited to just 2 values, and usually cannot describe extended sequential behaviors

In fact, similar questions about modeling adequacy have been asked about computerized reactive systems, concerning the ability of he model to faithfully capture the behavior of a continuous environment.

The Answer: A Hierarchy of Models



Cost of Higher Precision

As we proceed to more precise models, our ability to perform automatic analysis of models decreases significantly.

- For the Reactive-Systems model, it is possible to perform automatic verification of systems with up to several thousands boolean variables (signals).
- For the Real-Time model, it is possible to analyze systems with hundreds of variables and tens of clocks.
- For the Hybrid-Systems model, it is possible to analyze models with tens of variables.

Added Benefits of Formal Models

Besides providing a precise, unambiguous description of behaviors, reactive modeling also enables the additional functionalities of:

- Analysis (Verification) — Being able to formulate queries and find out whether a certain behavior is possible, or confirm that a certain class of undesirable behaviors is impossible. This functionality enabled the implementation of smart play-out in the play-in/play-out tool.
- Synthesis — Being able to construct a fully executable model from a set of reactive properties. This serves as the basis of automatic conversion from a set of LSC's into a Statechart specification.

These added functionalities proved very useful also in the biological context.

Differences in Objectives and Criteria

There are differences between the objectives expected from modeling of the artificial (e.g. design of computerized reactive systems), and that of the natural (such as biological modeling).

In modeling of natural phenomena, it is important to achieve:

- The ability to predict behaviors that have not been observed yet.
- The ability to explain the phenomena, and uncover hidden mechanisms and underlying principles.

In specification of man-made artifacts, important criteria are

- Abstraction and freedom from implementation bias

In both cases, it is important to achieve testability and succinctness.

Ability of Models to Explain

Often, a higher degree of “explainablity” is attained by the inclusion of non-observable elements. This often suggests an internal mechanism that can explain why the observed behavior is produced.

For example, the inclusion of pathways and some concrete representation of the propagation of inter-cellular signals in the C. Elegans.

A striking counter example is the work of Wolfram who manages to reproduce the behavior of an amazing diversity of processes, all by the use of cellular automata. This ability to emulate the external behavior, unfortunately, provides no explanation of the inner working and underlying principles of these processes.

Note that implicit unobservables are introduced even by purely behavioral formalisms such as LTL or LSC’s. These are the states attained after a partial behavior.

Pragmatics of Specifications

A great emphasis should be put on the pragmatics of specification. One may claim that usable specification and lack of a general understanding of how to use them is one of the main obstacles on the way to a wider adoption of formal methods in system development.

Because:

- A completely formally verified program is only as good as its specification. How do we validate the specification?
- Checking the specification for consistency and completeness.
- How do we elicit a formal specification from an "informal" user?
- Validation of a specification is necessarily an informal process, but may be greatly assisted by several different formal procedures (e.g. checking vacuity, etc.)

Multiplicity and Diversity of Specification Approaches

There are many different approaches to specification. Each associated with its own refinement and verification methods.

Some of this diversity can be justified by the application focus, for example:

- Specifications concentrating on control and behavior (reactive systems), vs.
- Data-centric specification methods.
- Architectural, structural and interconnection aspects.
- In concurrent systems, distinguish between systems communicating by shared variables and those communicating by message passing (synchronous or asynchronous).
- Model-based vs. Property-based specification styles

As long as we can provide explicit guidelines of when each specification style should be used, this proliferation is positive and mutually beneficial. Should take care of integration.

We should definitely beware of unjustifiable and inexplicable diversity.

A Property-Based System Development Project Prosyd

While the rest of the world is adopting a model based approach to system development, the European project Prosyd proposes a property based approach, for the systematic development of digital designs.

The gospel according to UML (Unified Modeling Language) and similar approaches is that system specification, which is the starting point in any development process, should be presented in terms of a model, i.e., an abstract program displaying the same external behavior as the expected system.

Property based design postulates that the specification should be presented as a list of properties, and any system satisfying these properties is a correct implementation.

The Pros and Cons of the Two Approaches

Property-Based specifications are strong on compositionality and modularity. It is easy to add, withdraw, or modify requirements.

More challenging are the questions of:

- Consistency. Making sure that the integration of the property set is consistent. Has a formal basis and available algorithms.
- Completeness. Making sure that we have not missed relevant properties. No direct formal support.
- Realizability. In the context of a reactive system, this requires a guarantee that the specification can be implemented by a functional module that need not predict a future behavior of the environment, such as
  $(\mathit{output} = 1) \iff \diamondsuit\, (\mathit{input} = 1)$.

For model-based specifications, changes are more difficult (non-modular). Consistency and Completeness are almost automatic by construction. However, compliance with the customer's expectations are as problematic as for any other approach.

Some More Distinctions

Typically, a model based specification is structured by modules. It tells the intra-object story. Because of that and other characteristics, it is closer to the implementation and may introduce implementation bias.

In contrast, a property based specification is structured by properties. It tells the inter-object story. It often provides a specification on a more abstract level.

Historical (and Personal) Perspectives

When LTL was first introduced, the methodology associated with it has been a property based one, by which a specification is comprised of a property list.

In a sequence of workshops, this methodology has been challenged and claimed to be non-scalable. Canonical examples have been shown to be specifiable by a model-based approach but not by a property based approach.

Typical software examples are serializability in data bases. Typical hardware examples are out-of-order execution.

Industry (software and systems) refused to adopt the property-based approach.

A Possible Compromise

One possible compromise was to distinguish between

- Requirement specification which is property based. One can use LTL (or its extension PSL) for presenting such a specification, and
- Executable system specification which is model based. One can use a language such as statecharts to describe such specifications.

A full development process for systems should start with a requirement specification, massage it into a system specification, which would then be transformed (perhaps even automatically) into a running implementation.

Some sectors of industry (specifically avionics and other safety critical sectors) were ready to embrace this general idea. However, at the beginning they were ready to pay only for the second layer.

Recent Trends I

Leslie Lamport, who was among the first to point out the shortcoming of the pure property based approach, stayed with linear TL. He uses his own version TLA to write model based specifications. Namely, to specify fair transition systems which is a universal modeling paradigm.

Recent Trends II

David Harel who is responsible for the development of the language of statecharts, one of the most prevalent model-based specification language, has recently shifted his attention to property-based specifications starting with the UML requirement language of message sequence charts.

Proposing the more expressive language of live sequence charts (LSC's), he is currently developing various tools which will support the construction of LSC-based requirement specification and its translation into an executable specification, or its direct execution.

The play-in/play-out paradigm and machinery address the important question of specification elicitation from a non-formal customer.

Property-Based System Design

While the rest of the world seems to be moving in the direction of model-based design (see UML), some of us persisted with the vision of property-based approach.

Specification is stated declaratively as a set of properties, from which a design can be extracted.

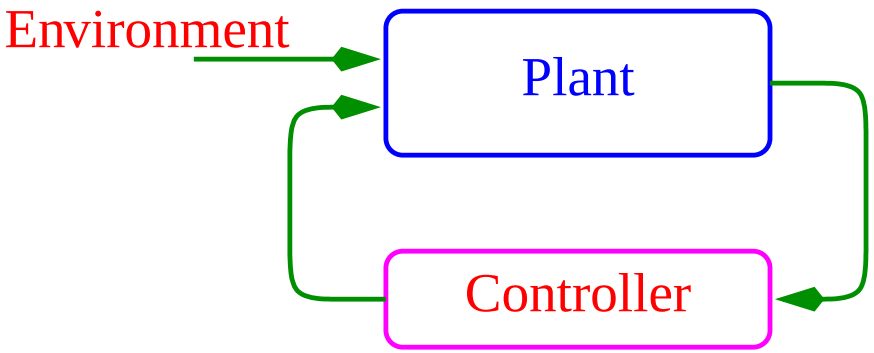
This has been studied in the hardware-oriented European project PROSYD.

Design synthesis is needed in two places in the development flow:

- Automatic synthesis of small blocks whose time and space efficiency are not critical.
- As part of the specification analysis phase, ascertaining that the specification is realizable.

The Control Framework

Classical (Continuous Time) Control



Required: A design for a controller which will cause the plant to behave correctly under all possible (appropriately constrained) environments.

Discrete Event Systems Controller: [Ramadge and Wonham 89]. Given a Plant which describes the possible events and actions. Some of the actions are controllable while the others are uncontrollable.

Required: Find a strategy for the controllable actions which will maintain a correct behavior against all possible adversary moves. The strategy is obtained by pruning some controllable transitions.

Application to Reactive Module Synthesis: [PR88], [ALW89] — The Plant represents all possible actions. Module actions are controllable. Environment actions are uncontrollable.

Required: Find a strategy for the controllable actions which will maintain a temporal specification against all possible adversary moves. Derive a program from this strategy. View as a two-persons game.

Recording of Experiments as General Rules

An important part of the development of the C. Elegans model was the transcription of reported experiments into universal LSC's. A major problem is that the experiment usually reports about the elements that were modified from their natural "wild-type" status. The question is how to represent the requirements about the unmodified elements.

A Motivating Example

Assume that we have three possible genetic factors which are represented by the boolean variable x_1, x_2, x_3, where $x_i = 1$ means that the corresponding factor has been mutated.

Assume also that results of experiments are represented by the value of the phenotype variable $y \in \{1, 2, 3\}$. Consider the following two experiments:

	x_1	x_2	x_3	y
Experiment 1	M	W	W	1
Experiment 2	W	M	W	2

We would like to enter this information into a rule that will enable us to predict the outcome of an experiment in which both x_1 and x_2 are mutated.

- First attempt. Model $\begin{cases} x_1 = 1 \wedge x_2 = 0 \wedge x_3 = 0 & \rightarrow \quad y = 1 \\ x_1 = 0 \wedge x_2 = 1 \wedge x_3 = 0 & \rightarrow \quad y = 2 \end{cases}$

  Does not support prediction. The case $x_1 = 1 \wedge x_2 = 1$ does not imply anything.

- Second attempt. Model $\begin{cases} x_1 = 1 & \rightarrow \quad y = 1 \\ x_2 = 1 & \rightarrow \quad y = 2 \end{cases}$

  Inconsistent!. The case $x_1 = x_2 = 1$ implies $y = 1 \wedge y = 2$

Have to reconcile the conflict between high predictability and inconsistency.

Introducing Don't Care Variables

Propose a general model of the form:

$$x_1 = 1 \ \wedge \ x_2 \leq d_1^2 \ \wedge \ x_3 \leq d_1^3 \quad \rightarrow \quad y = 1$$
$$x_1 \leq d_2^1 \ \wedge \ x_2 = 2 \ \wedge \ x_3 \leq d_2^3 \quad \rightarrow \quad y = 2$$

A value $d_i^j = 1$ means that in experiment No. i, we do not care about the value of x_j. The value $d_i^j = 0$ means that in experiment No. i, the value of x_j must be 0 implying that this factor must be wild type.

The mathematical problem we try to solve is:

> Find a solution with a maximal number of $d_i^j = 1$ such that, for any combination of $x_j's$, the set of implications is consistent.

For example, in the motivating example, we can take $d_1^2 = d_1^3 = d_2^3 = 1$ and $d_1^2 = 0$. Then the new experiment $x_1 = x_2 = 1$, $x_3 = 0$ yields the prediction $y = 2$.

Approaches to a Solution

Consider a given set of experiments. Let ϕ be the formula representing the conjunction of their implications where, for each experiment i and genetic factor x_j, we include $x_j = 1$ whenever x_j has been mutated in this experiment, and include the terms $x_j \leq d_i^j$ if x_i has been left wild type.

A given assignment of boolean values to d_i^j, leads to a consistent model iff the following formula is satisfiable:

$$\forall x_1, x_2, \ldots, x_n \; \exists y_1, y_2, \ldots, y_m : \phi$$

We can find a solution with $k \geq 0$ positive d_i^j iff the following formula is satisfiable:

$$\Psi(k): \quad \exists d_1, \ldots, d_p [(d_1 + \cdots + d_p = k) \; \wedge \; \forall x_1, x_2, \ldots, x_n \; \exists y_1, y_2, \ldots, y_m : \phi]$$

For a given k, the satisfiability of $\Psi(k)$ can be checked, either by BDD's (we used the tool TLV), or by a QBF solver (such as Quaffle).

To find an optimal solution for a maximal k, we can use binary search on the value of k.

We refer the reader to the paper in TACAS'07, where more efficient approaches to the solution of this problem are elaborated.

Conclusions

We strongly believe that the methodology of reactive specifications and analysis will prove very beneficial to Biology.

- More Biological-User friendly versions of the methodology and corresponding tools should be developed.
- Better approaches to the mutual refinement of reactive models and their validation by experiments should be studied and implemented.
- Computer Scientists should study the Biological applications and enhance their thinking and methods to reflect the special needs of Biologists. Biological adoption of the reactive methodology, can serve as a superb wet lab for for further development of our ideas and methods.