By permission of the Royal Society this paper is reproduced from *Phil. Trans. R. Soc. A* (2008), **366**: **From computers to ubiquitous computing, by 2020**.

Understanding Ubiquitous Computing

How can you hope to understand a ubiquitous system, whether you are a user embedded in it, or an engineer building it, or a scientist analysing it? Never before have such huge systems be envisaged. We have to lift the scientific status of informatics to provide this understanding. It can only be done by building a coherent **tower of models**. We cannot and need not apply uniform constraints to the nature of models; they must cope with the enormous variety of concepts used to describe ubicomp systems—from quasi-human properties like goals, self-awareness, reflectivity, negotiation at the top of the tower to formal machines at the bottom.

When we build a model, it is to *explain* something: either a natural phenomenon as in natural science, or a physical artefact as in engineering. Or it may explain another model. This last case applies especially in informatics; our ultimate real artefacts are so complex that they can only come about through several levels of modelling. Hence 'tower of models'.

In natural science the reality precedes the model; in informatics the model often comes first, e.g. it is a specification. But the *notion* of model is the same. In each case realities live at the base of the model tower; we can think of a reality as an extremal model that explains nothing else.

Realities may be heterogeneous; e.g. to model aircraft flight we model *natural* phenomena—the weather—by meteorology, and the artefact—the plane—by combining an electro-mechanical model with embedded software. Combination of models reflects combination of realities. The plane's software is itself a model (of circuit diagrams and their behaviour, in turn realised by an initialised computer). French informaticians recently explained the software for the Airbus at one level higher in the tower—by logical formulae. The explanation was evidenced by *abstract interpretation*.

So often one model A *explains* or *specifies* another, i.e. B; in turn we say B *realises* or *implements* A. These different verbs denote the same relationship. This relationship must be *evidenced*. If B is a reality, the evidence is by observation. When both A and B are models, the evidence may even be a mathematical proof. Such a proof can be by human or machine—in the latter case perhaps by *model-checking*.

It follows, inescapably, that a model has two parts:

(i) a class of entities (e.g. syntactic forms), and

(ii) how these entities behave.

Then, when A is realised by B, the realisation is evidenced by an argument showing how the behaviour of B-entities correctly matches that of the A-entities they realise. A model need not be formal for this to be done; a beautiful informal model is the report that described ALGOL60 nearly 50 years ago, and its implementation by a stack machine can be evidenced by careful informal reasoning.

Ubicomp will be a nightmare unless a tower of models is, by increments, built and used. Conversely, in building it we seize the chance to justify the claim that—against public perception—there is deep scientific content in understanding software.

Robin Milner, *University of Cambridge, UK*