Modularity, redundancy and degeneracy: cross-domain perspectives on key design principles

Chih-Chun Chen
University of Cambridge
Department of Engineering
Trumpington Street
Cambridge, CB2 1PZ
United Kingdom
Email: ccc54@cam.ac.uk

Nathan Crilly
University of Cambridge
Department of Engineering
Trumpington Street
Cambridge, CB2 1PZ
United Kingdom
Email: nc266@cam.ac.uk

***Abstract*—This article reviews and consolidates different treatments of modularity, redundancy and degeneracy across different disciplines and different system types (spanning both Systems Science and Systems Engineering). This is done with two main objectives. The first is to draw out and compare the perspectives and classifications that are available for these important concepts. The second is to point to areas where there has been a lack of precision or where more rigorous distinctions would help make discourse on these concepts more productive. For modularity and redundancy, we first consider the definitions and variants associated with these concepts. We then identify some key classificatory principles which might be used to distinguish between variants so as to illustrate the diversity across domains and contexts. Degeneracy is addressed by explicitly relating it to redundancy and modularity. We identify two key areas where lack of explicit treatment often results in confusion for the concepts of modularity, redundancy and degeneracy: (i) the level (or 'granularity') of analysis; (ii) whether it is types or instances that are being referred to. We hope that by making readers aware of the confusions that can arise, it will also encourage more serious dialogue to take place between practitioners and researchers in different domains so that existing bodies of knowledge can be usefully shared.**

I. Introduction

Modularity and redundancy are concepts that prevail in both Systems Engineering and Systems Science domains. For those looking to create or understand systems that perform well in changing circumstances, these two general concepts relate to useful design principles or explanatory frameworks. Modularity and redundancy can be observed in a diverse range of application domains, from Manufacturing, to Software Engineering, to Systems Biology,[1] yet there is little useful dialogue *between* domains on how these concepts are defined, how their variants can be classified and how their associated costs and benefits can be balanced. Within any single discipline, modularity and redundancy are also typically discussed independently of each other even though they can be seen as related and even though consideration of one concept is aided by consideration of the other. Another concept that is related to both modularity and redundancy is degeneracy, which has been much-cited as a characteristic of biological systems which perform well in changing circumstances. Within Biology, degeneracy is often discussed with respect to redundancy but those interested in redundancy in other domains seldom refer to degeneracy, despite its relevance and potential application.

There may of course be good reasons for the separate consideration of modularity, redundancy and degeneracy, and good reasons for the separate treatment how each can be identified or applied in different domains. Each of these three terms can be interpreted as referring to quite different things as we move from one domain to another or at the very least, and the domains differ in the focus they give to different aspects of these terms. For example, in Systems Engineering, the focus is often on constructing systems with modular or redundant architectures so as to attain desirable system properties such as reliability, robustness and scalability (sometimes collectively known as the -ilities), e.g. [47], [53]. Systems Sciences, on the other hand, tend to focus on the emergence of modularity and redundancy from the interacting base level entities that make up the system, e.g. [36], [37]. For example, in society, independent functional units can be observed to form spontaneously from the interactions of autonomous individuals.

Despite differences between disciplines, there are also substantial, *and growing*, similarities in the systems they consider and the ways in which they consider them. Emerging and converging technologies blur the boundaries between previously distinct types of system and the concepts, principles and practices associated with them. There are many examples of this: modern computational systems and the internet are studied as natural ecologies [35], [40]; evolutionary programming and evolutionary electronics study the way selection and diversification (inheritance) mechanisms in different environmental conditions (fitness landscapes) give rise to differences in the space of design solutions [11], [25]; complex sociotechnical systems are viewed as partially designed and partially evolving [26]; bioengineering seeks to design artificial systems from biological substrates [29]. At the same time, systems-based approaches to solving problems require novel practices for 'engineering' systems with large degrees of freedom [9] by intervening at the level of base components. To this end, engineering practices can make significant contributions to the Systems Sciences by providing disciplined methods for analysing systems [34], [2]. As previously distinct areas of theory and practice converge, the key concepts and terms that

[1] Concepts of modularity and redundancy have also been applied to a wide range of other systems, including organisations [9], economies [9], cognitive systems and the mind [31], natural language [57] and various cultural artifacts, such as educational curricula, literature, music, sports and law [12].

978-1-4799-2086-0/14/$31.00 ©2014 IEEE

they share form important connections between the relevant communities of interest. As such, understanding how modularity, redundancy and degeneracy are conceptualised across these different applications can help us identify the opportunities for cross-domain knowledge exchange.

To promote exchange of knowledge across domains, this paper reviews and relates the concepts of modularity, redundancy and degeneracy across different system types. For both modularity (Section II-A) and redundancy (Section II-B), we first consider the definitions and variants (e.g. *types* of modularity) associated with these concepts. We then identify some key classificatory principles which might be used to distinguish between variants so as to illustrate the diversity across domains and contexts. Section II-C addresses degeneracy by explicitly relating it to redundancy and modularity, revealing the need for more precise consideration of concepts such as 'function', 'system' and 'level'. These ideas are relevant to a broad range of application areas but for brevity, we primarily draw our examples from only a subset of these, emphasising digital, mechanical and biological systems. Making statements across these and other domains sometimes requires the use of abstract language. However, the examples offered and the diagrams presented are intended to permit translation to specific domains. From our survey and synthesis we identify key distinctions that should be made with respect to the three concepts so as to permit more productive discussions between and amongst practitioners, designers and scientists.

II. Modularity, Redundancy and Degeneracy

Modularity, redundancy and degeneracy can each be defined in terms of how systems are composed and how they behave. The composition of a system can defined by how functions are assigned to components and how those components relate to each other. The behaviour of a system can be defined by the way in which the system permits or tolerates change. This talk of *systems* and their *functions* highlights an important feature of modularity, redundancy and degeneracy: they are determined by perspective and purpose. Firstly, what defines a system, its boundary and therefore its components and environment is at the discretion of the observer. Systems can be viewed at different levels of abstraction and whether something is viewed as a component of a system or as a system itself varies from observer to observer [17], [43]. Secondly, unlike system properties and behaviours, which are observed, functions are assigned to systems by people. Systems do lots of different things and defining which of these things are functions is a matter of perspective [23].

The perspective-dependent nature of system boundaries (i.e. what counts as component, system, and environment) and functions makes concepts that rely on them (like modularity, redundancy and degeneracy) perspective-dependent. It is important to remember this when considering these concepts, especially when comparing work from different authors or different fields as we do below. It should also be stressed that when we use terms like 'components' (or 'systems' or 'environments'), we do not make any particular assumptions about the substrate or media from which those entities are composed. While it is most intuitive for some people to think of components as physical entities, we might also treat data, time or processes as components of a system (and those components might make that system more or less modular, redundant or degenerate). For example, we might have a software system with time redundancy (reprocessing when there is available time) or a barcode with information redundancy (using more space to duplicate a section of the code). Similarly, when we use the term 'structure', we include the organisation of non-physical entities such as processes and data.

A. Modularity

The concept of modularity relates to the independence of components, which are thus called *modules*, and the interchangeability of modules that this independence permits. Independence and interchangeability can be achieved in different ways and there are consequently different ways to define modules and modularity. Some definitions emphasise *interactions*, with weak interactions between modules and strong interactions within modules [7]. [2] Some definitions emphasise *interfaces*, with well-defined interfaces between modules, and between modules and other parts of the system [65], [54]. Still other definitions emphasise *functions*, with individual modules performing distinct functions rather than functions being shared across multiple components [66].

Over the years, scholars and practitioners have identified different types of modularity and a number of classifications have been suggested. Gathering such classifications reveals that these are often domain-specific taxonomies of observed practice rather than formally constructed typologies that might apply across domains.[3] In particular, the basis upon which the different types of modularity have been distinguished are seldom stated and are sometimes mixed (see Table I).[4] Nevertheless, these classifications or some combination of them, are frequently cited in the modularity literature (e.g. [54]) and are a useful way of illustrating the various ways in which modularity can be manifested.

Terms such as 'encapsulation', 'information-hiding' and 'blackboxing' are used to refer to the fact that the internal structure of a module can be decoupled from the way it interfaces with other components and hence how it relates to the system as a whole. The term 'component family' or 'module type' can be used to refer to the set of components that can be substituted for one another due to the fact that they share a common interface (compatibility with other components).

Standardisation of component interfaces has different implications for how components can be combined. On the one

[2]This definition of modular systems is similar to some definitions of systems themselves, where system boundaries are often chosen to maximise the ratio of within-boundary interactions to across-boundary interactions [55], [59]. It is also the assumption underlying quantitative modularity measures based on correlation, covariance or information entropy (e.g. [63], [68]).

[3]Classification theory distinguishes between conceptually-derived and empirically-derived classifications, referred to respectively as typologies and taxonomies in [6], but terminology varies between authors. Along with other classification theorists [14], [44], Bailey proposes two rules of classification: (1) that the classes formed should be both exhaustive (i.e. everything is classified); and (2) that the classes should be mutually exclusive (things only appear in one class).

[4]Although the typologies in Table I tend to originate from a specific Systems Engineering domain, e.g. Manufacturing, we are also aware of various attempts to generalise these to other domains. For example, the typology defined in [65] was initially formulated with the manufacture of products in mind, but has more recently been applied to services [15], also exemplifying the application of the term 'components' to non-physical entities.

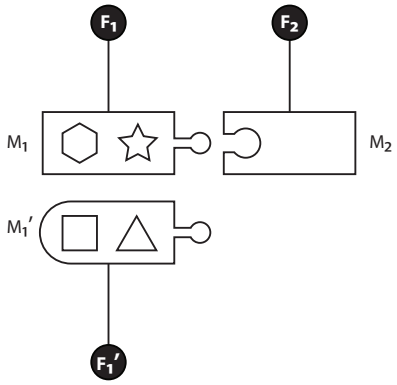

Fig. 1. General representation of modularity. Module M_1 (with the hexagon and star components) and module M_1' (with the square and triangle components) are variants of the same component family (module type) and may satisfy different functional requirements F_1 and F_1' respectively. M_1 and M_1' have the same interface, which allows them to connect with module M_2 (which performs function F_2) but the constituents or internal structure of module M_1 and M_1' might differ from each other and to M_2. M_2 only has one variant, which means that it is common to all system variants and can be seen as a standard component which can connect to either M_1 or M_1'.

hand, standardisation permits a given component to be used in different systems (sharing components between systems). On the other hand, with respect to a particular system, different components of the same component family might be substituted for each other (swapping components in and out of the same system). This substitution can either permit different functional requirements to be fulfilled through the manifestation of different properties and behaviours, or it can permit the same functional requirements to be fulfilled through different means which can be substituted for one another. [5] In designing a system with modularity, architectural decisions therefore have to be made about which components are variable and which are common to different system variants.

Throughout a system's lifetime, controlling components' interactions, interfaces and functions generally allows for the standardisation of components and the potential for variations in the system through combining those components in different ways. This allows the system to exhibit desirable lifecycle properties such as flexibility, extensibility and interoperability [66], [7], [33]. A modular system architecture also influences, in a similar way, the other processes and structures that are associated with it, including the design process, the supply chain and their associated economies [7], [66], [56]. These benefits of system modularity can come at a cost however, and systems that are designed only for technical performance will generally exhibit more integral (less modular) architectures [38], [10], most likely because components are not fully optimised for the system in which they operate. There can also be wider implications for innovation due to resistance to architectural change [16].

When analysing a system's modularity, how we choose to decompose the system is important. This is because different decompositions result in different 'levels',[6] which treat different sets or groups of constituents as components, which can result in very different architectures (and hence modules) [21], [54]. With respect to modularity in Systems Engineering, various attempts have been made to more systematically describe different levels [52], [48]. For example, in [48], the authors identify four levels of modularity: the component level, the module level, the sub-system level, and the system level. Decomposing a system thus requires decisions to be made over which constituents should be grouped together to form the lowest level 'components' (the use of this term by the authors is different from our own), which lowest level 'components' should be grouped together into modules, and so on. In Figure 1, for example, we might see M_1 and M_1' as representing the modular level with respect to their 'components' (e.g. the hexagon and the star), but with another decomposition, these 'components' might themselves be considered modules or subsystems.

Explicit recognition of different levels of analysis is also crucial in the Systems Sciences, where the issue is often how to decompose the system into modules with respect to particular functions (e.g. in Network Analysis [49], Systems Biology [50], Ecology [37], [32], Neuroscience [18]). Inferring modular *models* of a system that account for its observed properties or behaviour often involves detecting or modelling the emergence of higher level structures[7] and asking how, given the interactions going on between lower level components in the system (be they cells, animals, people or organisations), we get higher level structures, which in turn can be treated as encapsulated units and related to each other.

Another important application of 'level' is that it can be used to specify *temporal* as well as physical boundaries. A given set of constituents may be labelled a module at a given point in time or through a time range, but be seen as a transient state or subsystem at another. Temporal scope and resolution are also crucial in discussions of redundancy and degeneracy.

B. Redundancy

The concept of redundancy relates to the provision of additional capacity in a system so that system performance is maintained despite partial system failure. Just as with modularity, redundancy can be defined in different ways depending on how the additional capacity is considered. Some definitions emphasise *duplication*, with additional identical components included in the system to provide spare capacity. Some definitions emphasise *substitution*, with systems incorporating additional and different different means to achieve the same function.

Redundancy, like modularity, can be defined in terms of how functions are mapped to components and how the system behaves in the event of change. In the design and engineering literature, the changes of interest are typically component failures, and redundancy is thus discussed in terms of safety, reliability and robustness. If a component is assigned a function and if that component fails then some additional component will be required to perform the function (e.g. extra engines on an aircraft). Providing that additional component in the design of the system at the outset is an implementation of redundancy; the additional component is redundant until a failure occurs.

[5] The existence of these alternative components in the same system might also be a means of realising redundancy, as discussed in Section II-B.

[6] These levels are sometimes also referred to as 'granularity' [21], [4].

[7] Again, the term 'structure' is meant in the most abstract sense and can relate units that are physical or behavioural.

Classification criterion	Types
Component role in the system	**Basic** and **auxiliary** modules implement functions that are common throughout the product family (i.e. all systems of a given type) [51]. **Special** modules implement complementary and task-specific functions that do not need to appear in all the product variants. **Adaptive** modules implement functions related to the adaptation to other systems and to marginal conditions [51]. **Non-**modules are designed specifically for specific tasks [51]. *Modules in Biology can differ in their specificity and hence might be characterised as more or less 'adaptive' or in the extreme case 'non-modules'. They can also differ in terms of their necessity for the organism.*
Component assembly and interface compatibility	**Component swapping** modularity: product variants are obtained by swapping one or more components on the common product body [65]. **Component sharing** modularity: the same basic component can be used in different products [65]. **Fabricate-to-fit** modularity: product variants are obtained by combining one or more standard components with one or more components that can be produced with a continuously variable feature, such as length [65]. **Bus** modularity: product variants are obtained by providing a continuous common interface that can accept any selection of components from a set of component types with a compatible interface [65], [66]. **Sectional** modularity: product variants are obtained by mixing and matching in an arbitrary way a set of components as long as they are connected to each other at their common interfaces [65], [66]. **Slot** modularity: interfaces between different types of components are different [66]. **Combinatorial** modularity: product variants are obtained by differently combining the components of different component families. [54] adds combinatorial modularity to the above six variants distinguished in [65], [66]. *In Biology, developmental modules corresponding to physical structures or signalling networks in the organism are subunits that can be found in the embryo which develop into distinct functional elements of adult morphology [13]. These might be found across different but related species and exhibit the same behaviour [13] (analogous to 'component-sharing') or they can exhibit different behaviours in different species [24] (analogous to 'combinatorial modularity'). In Molecular Biology, domains of protein molecules can be seen as interfaces and hence determine the interactions and 'compatibilities' between molecules to perform some function [2].*
Lifecycle stage or interaction with the system environment	**Assembly modules** are components, or groups of components, that are assembled in clearly distinct stages to ease production [45]. (In [5], a product-oriented view of modularity distinguishes different types of modularity according to the capabilities they afford at different points of the product lifecycle. **Manufacturing** modularity: permitting a variety of products to be assembled from a limited number of standardised parts [45]. **Product use** modularity: permitting customisation during product use [5]. **Limited life** modularity: permitting expired (worn out or depleted) components to be replaced [5]. **Data access** modularity: permitting storage and transfer of information separate from the rest of the system [5]. Similarly, in [39], a distinction is drawn between *function-oriented* modules (which include basic, auxiliary and adaptive modules), and *production-oriented* modules, where design is based on production considerations alone, and in [7], the authors distinguish between modularity in design, production and use respectively.) *In Systems Biology, we can distinguish between **Variational** modules: features that vary together but are relatively independent of other such sets of features [68]; **Developmental** modules: parts of an embryo that are quasi-autonomous with respect to pattern formation and differentiation, or an autonomous signalling cascade [68] and **Functional** modules: features that act together in performing some discrete physiological function that is semi-autonomous in relation to other functional modules [68].*

TABLE I. TYPES OF MODULARITY IN SYSTEMS ENGINEERING BASED ON DIFFERENT CLASSIFICATION CRITERIA WITH ANALOGOUS DISTINCTIONS IN SYSTEMS BIOLOGY TO ILLUSTRATE THE PARALLELS THAT CAN BE DRAWN.

Fig. 2. Generic representation of redundancy. Left: The function F can be realised by two (or possibly more) components which either share the load *synchronously* (as shown here), or realise the function independently and *asynchronously* (with one being used whilst the other is spare). Right: When one of the components fails (as indicated by the X), the function is realised by the other component (as shown here), or the function 'switches' from one of the components to the other.

Many implementations of redundancy reduce the modularity of the system because the mapping between functions and components becomes more complex (when considered *through* time rather than at a given time).

In the Systems Sciences, discussions of redundancy tend to focus more on understanding how functions can be realised in different ways by the same system. In this context, the assumption is that at least some of the functions are distributed across several components. These components might be from the same 'component family' (e.g. cells in a tissue) or from different 'component families' (e.g. cells in an organ).[8]

Again, as with modularity, scholars and practitioners have identified different kinds of redundancy and various taxonomies have been proposed. In Systems Engineering, redundancy discussions tend to centre around which components to make redundant, how much redundancy there should be, and what form the redundancy should take so as to optimise for factors such as reliability, cost and performance [17], [46]. TableII shows several typologies based on different (often implicit) classificatory criteria.

In addition to the redundancy classification criteria in Table II, we can distinguish between different system behaviours during and after component failure. These behaviours range from instant recovery so that the impact on performance is negligible (e.g. load balancing, hot spare) to noticeable degradation in performance while remaining within the bounds

[8]Of course, as with modularity, the level at which one defines both components and functions determines the component family. For example, the functioning of an organ is distributed amongst cells of different types, with the population of cells of each type playing a different role in realising the function (and therefore having different compatibilities with the rest of the system), while a tissue's function is distributed amongst cells of the same cell type.

Classification criterion	Types
Execution mechanism for fulfilling required function	**Principle redundancy**: multiple non-identical components operating on different principles (possibly arranged in parallel) are capable of fulfilling the required function but do so by different means. Systematic failure of all components is less likely because each component is susceptible to different failure mechanisms [51]. (A parallel can be drawn between this and N-version programming [20].) **Selective redundancy**: the outputs of multiple active components are selected from [51] (possibly selecting any component that is operational). **Comparative redundancy**: the outputs of multiple active components are compared (possibly taking any variation in outputs to be negative) [51]. **Combinations** of the above (e.g. principle-comparative redundancy [51]. *In biological systems, it is more difficult to distinguish between cases where different components are realising a function at the same time with one component or set of components 'winning' (as in selective redundancy) and cases where a component or set of components realises the function by default but others can be substituted for it (as in principle redundancy).*
Role of redundant modules during normal operation and during failure	**Active redundancy**, **Load balancing**: multiple components (possibly identical) actively share a function during normal operation. Failure (e.g. of a single component) increases load on the remaining components [30], [51] (in [58], this is 'parallel' redundancy, which is distinct from 'parallel' redundancy in [51]) **Partial active redundancy**: all components are normally working but the system will continue to function satisfactorily provided that a certain number of components of the system continue work [62]. **Passive redundancy**: spare components (possibly identical) are inactive during normal operation and only implemented in case of active component failure. Failure may result in a change of performance (if the spare components are not identical) and may lead to system down-time (if the spare component is not immediately available - see hot and cold spares below) [51] (in [58], this is 'serial' redundancy, which is distinct from 'series' redundancy in [51]).
Recovery time and availability of extra components (for passive redundancy)	**Cold spare/standby**: redundant component needs to be switched on or initialised when there is failure (in the case of computer clusters, this might require installing and configuring an extra node) [67]. **Hot spare/standby**: redundant component ready to operate in failure (but does not participate in the operation of the system when it is functioning normally, in contrast to load balancing.) [67] **Warm spare/standby**: redundant component is in some intermediary state of readiness (in the case of computer clusters, this might entail starting software on the extra node and ensuring data is up to date) [67].

TABLE II. TYPES OF REDUNDANCY IN SYSTEMS ENGINEERING BASED ON DIFFERENT CLASSIFICATION CRITERIA WITH ANALOGOUS DISTINCTIONS IN SYSTEMS BIOLOGY (IF APPLICABLE) TO ILLUSTRATE THE PARALLELS THAT CAN BE DRAWN.

of acceptability (e.g. warm spare, cold spare). In [37], the term 'soft redundancy' is also introduced to describe systems in which different components or mechanisms provide a function over different levels of demand but where these components or mechanisms overlap for a given level of demand. For example, Holling cites the example of temperature regulation in endotherms, which can be realised via five distinct mechanisms, from evaporative cooling to metabolic heat generation and where more than one of these mechanisms can provide temperature regulation for a particular set of conditions. Soft redundancy is observed in natural, self-organising systems in which the system itself adapts.

Error reporting can be seen as an auxiliary behaviour which can have implications for how the rest of the system responds to the failure, e.g. error handling behaviour can adapt to the type of error detected [22]. However, some authors also use the error-reporting behaviour (or absence of such behaviour) as a basis for classification (e.g. In [41], the authors distinguish between 'active' hardware redundancy, where errors are detected and reported and 'passive' hardware redundancy where they are masked). In this paper, we regard error handling behaviour as an issue that is separate to redundancy even though different redundant architectures may be more or less amenable to different types of error response.

C. Degeneracy

Although modularity and redundancy are near-ubiquitous concepts in Systems Engineering and Systems Science, the related concept of degeneracy is much less well known. Degeneracy refers to the ability of structurally different components to perform the same function so that the failure (or absence) of a critical component is compensated for through *widespread* compensatory adjustments elsewhere in the system [64], [28], [69]. This has two aspects to it. One is that components which are degenerate with respect to each other will appear functionally equivalent in one set of conditions (and hence give rise to redundancy since they both map to the same function in this set of conditions). The other is that these same components will be functionally distinct under another set of conditions thus masking any potential redundancy in the system. Note that whether degeneracy is a type of redundancy or distinct from redundancy depends on whether redundancy is defined only in terms of duplication. Biologists typically conceive of redundancy as an engineering concept requiring the of duplication of identical components and they thus distinguish degeneracy from redundancy on that basis.

In [69], the authors note two main benefits that degeneracy can confer on a system, both arising from the increased complexity it gives to the system. Firstly, it can be a source of robustness, since functions can be fulfilled in multiple ways. As noted in [51], this renders systemic failure less probable. Secondly, degeneracy can lead to increased evolvability since different realisations of a function are likely to have different sensitivities to different selection pressures. Although robustness and evolvability are considered in Systems Engineering, to date, references to degeneracy are most commonly found in discussions of self-organising, complex adaptive systems (typically biological). The concept has received less attention in discussions of designed systems, where duplication of components is the standard approach to providing robustness against component failure [60]. However, architectures with principle redundancy or selective redundancy [51] (see Table II) can also be said to be degenerate since the same function can be fulfilled by components of different types. However, a distinction can be drawn between systems that have components that are dedicated to performing a function even if

they are spare (i.e. redundant systems) and systems that have components that are dedicated to other functions but that can be 'repurposed' as required (i.e. degenerate systems).

Degeneracy is related in intricate ways to both modularity and redundancy. With respect to modularity, degeneracy can reduce module integrity or purity (i.e. simple component-function mapping) if component-function mapping is not retained through the system's lifetime (see Figure 3). In this case, the mapping between functions and components so that components perform different functions at different times. However, at a given point in time or within a given time range, degeneracy in a system need not reduce system modularity because within this temporal scope, the mapping between functions and components may not change. With respect to redundancy, degeneracy is a mechanism for realising a function in multiple ways, and these multiple 'routes' or 'paths' provide surplus capacity in the system and/or robustness against component failure. The ways in which degeneracy is implemented could mean that different variants of redundancy are instantiated by a degenerate system (rather than in a modular fashion). For example, degeneracy might be active or passive, serial or parallel, hot or cold, etc. (see Table II). Therefore, although no classifications of degeneracy have been proposed (or found in preparing this paper), classifications of degeneracy might be derived by considering the existing classifications of modularity and redundancy. [9]

At the component level, we can see degeneracy as a means of defining a 'family' of structurally different variants which perform the same function. If these variants co-exist in the same system, then degeneracy provides redundancy since there are extra (structurally different) component instances which can take over from each other to fulfill the function. While in the biological context it is natural to assume that multiple instances of different variants co-exist in the system, the level of redundancy they provide with respect to the system's function depends on how many of the instances are 'available' at any given time (see Figure 3). On its own therefore, the fact that a function can be realised by different types of components does not imply redundancy when the system is in operation. Rather, redundancy is a function of the degrees of freedom available to the system given the distribution of different component types (i.e. the number of instances of each type) and the set of functions that need to be performed.

In natural, self-organising systems, having functions realised by components with different structures also usually implies that components are potentially multi-functional. Structural differences allow components to perform different functions in different contexts so that "while degenerate components contribute to stability under conditions where they are functionally compensatory, their distinct responses outside those conditions provide access to unique functional effects, some of which may be selectively relevant in certain environments" [69]. This introduces another dimension to the

[9]Even in biological systems, the discussions of degeneracy are relatively recent (most work in which it is explicitly mentioned was published after the year 2000, although this may be simply a terminological rather than conceptual shift), because biological robustness was presumed to originate in (duplication) redundancy. This in turn was based on a technological analogy [60] and can be seen as an example of where cross-domain typologies of key design principles would be productive.

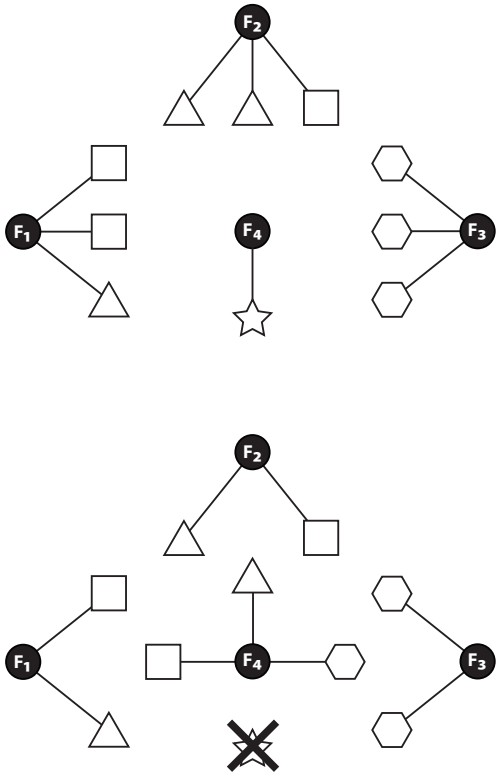

Fig. 3. General representation of degeneracy. Top: F_4 can be fulfilled by a single component of a given type (represented by the star). Bottom: in the event of component failure, F_4 can be shared by instances of three other types of components (represented by the triangle, square and hexagon) that are normally fulfilling other functions (here, F_1, F_2 and F_3)

relationship between degeneracy and redundancy in that the degree of redundancy might itself be context-dependent or incomplete under certain conditions (e.g. with degradation in performance, resulting in Holling's 'soft' redundancy). In other words, a given component can perform a particular function (or perform a particular function to a certain degree) under one set of conditions but perform a different function (or perform the same function to a lesser degree) under a different set of conditions. This also implies that the component can be characterised as a different module in different contexts and that if it exists in a system which realises these different contexts, it can no longer be treated as a 'pure' module but as a multi-functional component. Although it is most intuitive to associate the multi-functionality of components with real differences in their properties or behaviour under different conditions. However, this is not the only way in which components can be multi-functional. Instead, it may be that a component simultaneously performs different functions because of the different perspectives from which it can be viewed. For example, a traffic light has a signalling role at the level of individual drivers and a flow-control role at the level of the traffic system [23].

III. Conclusions and Further Work

From the discussions above, we identify two key areas that should be explicitly attended to to avoid confusion with the concepts of modularity, redundancy and degeneracy: (i) the level or 'granularity' of analysis, both for components and for

functions; and (ii) whether it is types or instances that are being referred to, both for systems and their components. As already discussed in Section II-A, different decompositions of the system treat different sets of constituents as components. A system with a modular, redundant or degenerate architecture at one level of decomposition may not have such an architecture at another level. In Systems Engineering, the notion of 'level' tends to be formulated in terms of components, functions and systems (see Section II-A). In the Systems Sciences, partly due to the influence of Physics, the notion of level is closely linked to observation. This leads to more generalised formulations which are typically based on 'scope' (the boundary of the system being considered) and 'resolution' (the distinctions that can be made within that boundary) [8].[10] Specifying the scope and resolution we are considering gives us a means of formally specifying the level or 'granularity' of analysis and tells us which entities we can treat as components and which we should treat as (sub-)systems.

Perhaps more subtle is the distinction between types and instances. A type, whether of a component or a system, defines a set of possible states, properties and behaviours that can be realised by its instances. However, any given instance of a type (at a particular time) will only realise a subset of these. For example, a system of a given type might be designed with redundancy, but this property may not persist through its complete lifespan; indeed, when considering such a system instance after there has been failure, there will no longer be redundancy until the redundant component(s) have been replaced. During the system's lifetime, modularity or redundancy in its architecture can confer different costs and benefits. For example, a redundant architecture may make a system's operation more costly (or less efficient) in a stable environment but allow the system to adapt at lower cost (and hence operate more efficiently) in a more dynamic environment. Similarly, when we make statements about naturally occurring 'types' or species (e.g. the degeneracy in human lymphocytes confers redundancy with respect to a given function), they may not hold for specific instances (i.e. degeneracy may not result in redundancy in a given human lymphocyte in a particular state at some point in time if all components are fully 'occupied' in different processes).

The possible conflations and confusions discussed above reveal the need for taking a more formal approach to defining modularity, redundancy and degeneracy. As well as permitting the exchange of knowledge across domains and applications, such a common formal framework would also allow us to quantify modularity, redundancy and degeneracy using existing measures (e.g. [64], [21]) and exploit more general optimisation frameworks to analyse the costs and benefits of different architectures (e.g. with respect to cost [54], flexibility [56], quality [3]). Equally important however, is the ability to map existing domain-specific classifications and typologies onto such a formal framework. It is hoped that the current article serves as a stimulus for cross-domain collaboration in establishing such a framework and mapping. Such a mapping would permit more efficient exploitation of existing bodies of knowledge across domains. This would, in turn, promote more effective approaches to understanding, maintaining and designing the many different types of systems that Systems Engineering and the Systems Sciences are concerned with.

[10]In [42], scope is also equivalent to α-composition, which refers to a set of components that are structurally related or integrated with each other, while resolution is equivalent to β-composition, which refers to a set of components that belong to a common category. With respect to the earlier modularity terminology, a 'component' can be seen to 'α-compose' and define a scope for its constituents (which might themselves be physical entities or even subsystems), while a 'component family' or 'module type' 'β-composes' its different variants.

ACKNOWLEDGMENT

This work was funded by the UK's Engineering and Physical Sciences Research Council (EP/K008196/1).

REFERENCES

[1] W. J. Abernathy and J. M. Utterback, "Patterns of industrial innovation," *Technology Review*, pp. 40–47, 1978.

[2] C. M. Agapakis and P. A. Silver, "Synthetic biology: exploring and exploiting genetic modularity through the design of novel biological networks." *Molecular BioSystems*, 5(7), pp. 704–713, 2009.

[3] B. Agard and S. Bassetto, "Modular design for quality and cost," in *Systems Conference (SysCon), IEEE International*, pp. 1–6, 2012.

[4] T. AlGeddawy and H. ElMaraghy, "Optimum Gradularity Level of Modular Product Design Architecture," *CIRP Annals, Design Track*, 2013.

[5] E. D. Arnheiter and H. Harren, "A typology to unleash the potential of modularity," *Journal of Manufacturing Technology Management*, 16(7/8), pp. 699–711, 2005.

[6] K. D. Bailey, *Typologies and Taxonomies: An Introduction to Classification Techniques (Quantitative Applications in the Social Sciences)*, SAGE Publications, Inc, 1994.

[7] C. Y. Baldwin and K. B. Clark, *Design Rules: The power of modularity*. MIT Press, 2000.

[8] Y. B. Yam, "A mathematical theory of strong emergence using multiscale variety," *Complexity*, 9(6), pp. 15–24, 2004.

[9] Y. Bar-Yam, *Making Things Work: Solving Complex Problems in a Complex World*, Knowledge Press, 2005.

[10] T. A. Bell, J. P. Jarrett, and P. J. Clarkson, "Exploring the Effects of Removing Process-Intrinsic Constraints on Gas Turbine Design," *Journal of Propulsion and Power*, 24(4), pp. 751–762, 2008.

[11] P. J. Bentley, *Digital Biology: How Nature Is Transforming Our Technology and Our Lives*, Simon & Schuster, 2002.

[12] J. G. Blair, *Modular America: cross-cultural perspectives on the emergence of an American way*, Greenwood Press, 1988.

[13] J. A. Bolker, "Modularity in Development and Why It Matters to Evo-Devo," *American Zoologist*, 40(5), pp. 770–776, 2000.

[14] G. C. Bowker and S. L. Star, *Sorting Things Out*. MIT Press, 1999.

[15] S. A. Brax and M. Toivonen, "Modularization in business service innovations," in *Proceedings of the XVIII ISPM Conference*, 20, pp. 103–119, 2000.

[16] S. Brusoni, L. Marengo, A. Prencipe, and M. Valente, "The value and costs of modularity: A Problem-Solvinng Perpective", *European Management Review*, 4(2), pp. 121–132, 2007.

[17] D. M. Buede, *The engineering design of systems: models and methods*, Wiley series in systems engineering, Wiley, 2000.

[18] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature Reviews in Neuroscience*, 10(3), pp. 186–198, 2009.

[19] A. Chambari, S. H. A. Rahmati, A. A. Najafi, and A. Karimi, "A bi-objective model to optimize reliability and cost of system with a choice of redundancy strategies," *Computers and Industrial Engineering*, 63(1), 2012.

[20] L. Chen and A. Avizienis, "N-Version programming: a fault-tolerance approach to reliability of software operation," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years*, IEEE, 1995.

[21] N. Chiriac, K. Holtta-Otto, D. Lysy, and E. S. Suh, "Level of Modularity and Different Levels of System Granularity," *Journal of Mechanical Design*, 133(101007), 2011.

[22] F. Cristian, "Exception Handling and Software Fault Tolerance," *IEEE Transactions on Computers*, C-31(6), pp. 531–540, 1982.

[23] N. Crilly, "Function propagation through nested systems," *Design Studies*, 34(2), pp. 216–242, 2013.

[24] S. Crosson, P. T. McGrath, C. Stephens, H. H. McAdams, and L. Shapiro, "Conserved modular design of an oxygen sensory/signaling network with species-specific output." *PNAS*, 102(22), pp. 8018–8023, 2005.

[25] K. A. de Jong, *Evolutionary Computation*, A Bradford Book, 2002.

[26] O. L. de Weck, D. Roos, and C. L. Magee, *Engineering Systems: Meeting Human Needs in a Complex Technological World*. MIT Press, 2011.

[27] C. E. Ebeling, *An introduction to reliability maintainability engineering*, McGraw-Hill, 1997.

[28] G. M. Edelman and J. A. Gally, "Degeneracy and complexity in biological systems," *Proceedings of the National Academy of Sciences*, 98(24), pp. 13 763–13 768, 2001.

[29] D. Endy, "Foundations for engineering biology," *Nature*, 438(7067), pp. 449–453, 2005.

[30] A. Ertas and J. C. Jones, *The engineering design process*. Wiley, 1993.

[31] J. A. Fodor, *The Modularity of Mind: An Essay on Faculty Psychology*. A Bradford Book/MIT Press, 1983.

[32] M. A. Fortuna, D. B. Stouffer, J. M. Olesen, P. Jordano, D. Mouillot, B. R. Krasnov, R. Poulin, and J. Bascompte, "Nestedness versus modularity in ecological networks: two sides of the same coin?" *The Journal of animal ecology*, 79(4), pp. 811–817, 2010.

[33] E. Fricke and A. P. Schulz, "Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle," *Systems Engineering*, 8(4), pp. 279–295, 2005.

[34] P. Fu, "A perspective of synthetic biology: assembling building blocks for novel functions." *Biotechnology journal*, 1(6), pp. 690–699, 2006.

[35] L. Gao, "On Inferring Autonomous System Relationships in the Internet," in *IEEE/ACM Transactions on Networking*, 2000, pp. 733–745.

[36] J. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[37] C. S. Holling, "Engineering Resilience versus Ecological Resilience," In *Engineering Within Ecological Constraints*, pp. 31–43, 1996.

[38] K. Hölttä, E. S. Suh, and O. de Weck, "Trade-off between modularity and performance for engineered systems and products," in *ICED 2005: The 15th International Conference on Engineering Design*, 2005.

[39] C-C. Huang and A. Kusiak, "Modularity in Design of Products and Systems," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28(1), pp. 66–77, 1998.

[40] S. Forrest, J. Balthrop, M. Glickman, and D. Ackley, "Computation in the wild," in *Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies*, E. Jen, Ed. Oxford University Press, pp. 207–230, 2005.

[41] B. W. Johnson, *The Design and Analysis of Fault Tolerant Digital Systems (Addison Wesley Series in Electrical and Computer Engineering)*, Addison-Wesley, 1989.

[42] J. Johnson and P. Iravani, "The Multilevel Hypernetwork Dynamics of Complex Systems of Robot Soccer Agents," *ACM Transactions on Autonomous and Adaptive Systems*, 2(2), 2007.

[43] M. W. Maier and E. Rechtin, *The art of systems architecting*, CRC Press, 2009.

[44] A. Marradi, "Classification, typology, taxonomy," *Quality & Quantity*, 24(2), 1990.

[45] D. A. McAdams, R. B. Stone, and K. L. Wood, "Functional Interdependence and Product Similarity Based on Customer Needs," *Research in Engineering Design*, 11(1), pp. 1–19, 1999.

[46] H. McManus and D. Hastings, "A framework for understanding uncertainty and its mitigation and exploitation in complex systems," *Engineering Management Review, IEEE*, 34(3), p. 81, 2006.

[47] H. McManus, M. Richards, A. Ross and D. Hastings, "A framework for incorporating "ilities" in Tradespace Studies," *AIAA Space*, 1, p. 941–954, 2007.

[48] J. H. Mikkola and T. Skjott-Larsen, "Supply-Chain Integration: Implications for Mass Customization, Modularization and Postponement Strategies," *Production Planning and Control*, 15(4), pp. 352–361, 2004.

[49] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, 103(23), pp. 8577–8582, 2006.

[50] D. Noble, "Modeling the Heart–from Genes to Cells to the Whole Organ," *Science*, 295(5560), pp. 1678–1682, 2002.

[51] G. Pahl and W. Beitz, *Engineering Design: Systematic Approach*. Springer-Verlag, 1996.

[52] L. Peters and H. Saidin, "IT and the Mass Customization of Services: the Challenge of Implementation.," in *Proceedings of the XVIII ISPM Conference*, 2007.

[53] A. M. Ross, D. H. Rhodes, and D. E. Hastings, "Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value," *Systems Engineering*, 11(3), pp. 246–262, 2008.

[54] F. Salvador, C. Forza, and M. Rungtusanatham, "Modularity, product variety, production volume, and component sourcing: theorizing beyond generic prescriptions," *Journal of Operations Management*, 20(5), pp. 549–575, 2002.

[55] M. A. Savageau, *Biochemical Systems Analysis: Study of Function and Design in Molecular Biology*. Addison-Wesley, 1976.

[56] S. B. Schapiro and M. H. Henry, "Engineering agile systems through architectural modularity," in *Systems Conference (SysCon) 2012*, IEEE, 2012, pp. 1–6.

[57] C. E. Shannon, "Prediction and entropy of printed English," *Bell Systems Technical Journal*, 30(1), pp. 50–64, 1951.

[58] D. Siemaszko and S. Pittet, "Impact of modularity and redundancy in optimising the reliability of power systems that include a large number of power converters," *Microelectronics Reliability*, 51(9-11), pp. 1484–1488, 2011.

[59] H. Simon, "The architecture of complexity," in *Proceedings of the American Philosophical Society*, 106, pp. 467–482, 1962.

[60] R. V. Solé, R. Ferrer-Cancho, J. M. Montoya, and S. Valverde, "Selection, tinkering, and emergence in complex networks," *Complexity*, 8(1), pp. 20–33, 2002.

[61] P. M. Swamidass, Ed., *Encyclopedia of Production and Manufacturing Management*, Springer, 2000.

[62] G. G. Thompson, *Improving maintainability and reliability through design*, Professional Engineering Publishing, 1999.

[63] G. Tononi, O. Sporns, and G. M. Edelman, "A measure for brain complexity: Relating functional segregation and integration in the nervous system," *PNAS*, 91, pp. 5033–5037, 1994.

[64] G. Tononi, O. Sporns, and G. M. Edelman, "Measures of degeneracy and redundancy in biological networks," *Proceedings of the National Academy of Sciences*, 96(6), pp. 3257–3262, 1999.

[65] K. Ulrich and K. Tung, "Fundamentals of Product Modularity," in *Winter Annual Meeting Symposium on Issues in Design Manufacture/Integration, ASME*, pp. 73–79, 1991.

[66] K. Ulrich, "The role of product architecture in the manufacturing firm," *Research Policy*, 24(3), pp. 419–440, 1995.

[67] H. Wada, J. Suzuki, K. Oba, "Modeling Non-Functional Aspects in Service Oriented Architecture," *Services Computing*, pp. 222–229, 2006.

[68] G. P. Wagner, M. Pavlicev, and J. M. Cheverud, "The road to modularity," *Nature Reviews in Genetics*, 8(12), pp. 921–931, 2007.

[69] J. Whitacre, "Degeneracy: a link between evolvability, robustness and complexity in biological systems," *Theoretical Biology and Medical Modelling*, 7(1), 2010.