Functional Concepts

	Arcadia		SYSMOD	
	System Mission	High-level goal to which the system should contribute Uses a number of system functions (regrouped in one or more system capabilities) to be fulfilled	*System Objectives*	Represent the main objectives of the vendor or owner of the system Used to understand the rationale of the requirements and to communicate the System Objectives to the developers of the system model Can be applied on higher-levels than Arcadias System Missions (see also Arcadia Operational Analysis level)
Differences	*System Capability*	Expected ability to supply a service contributing to fulfilling one or more missions Used to describe the system usage context Set of functional chains an scenarios that it references, and which more precisely describe the conditions for performing the system functions that contribute to it	*System Use Cases*	Services provided by the system to its system actors Provides a view on the system functions from the perspective of the system actors and describes the purpose of the system functions as well as the supporting functions Described and detailed by the Use Case Activites
Differences	*Not in Arcadia*		*Continuous Use Case*	It represents a continuous behavior of the system The trigger can be an internal event like a system state switch instead of an external trigger initiated by an actor. The result is typically a continuous output, such as compliance with a condition.
	Functional Chain	System behavior in a particular usage context to contribute to one or more system capability Decribes a path / ordered set of functions and functional exchanges that link them Control nodes can be defined (OR, LOOP, etc.) but should be seperated No encapsulation mechanism	*Use Case Activities*	Specifications of the system functions represented by the System Use Cases Specify the system functionality from the requirements perspective Describes the functional decomposition of the Systems Use Cases, for example as an activity tree (functions and sub-functions) Single functions of a System Use Case, their order of execution and the flow of objects Focus on Object and Control Flow Encapsulation mechanism used for functional hierarchy and decomposition
Differences	*Functional Exchange*	Reflects functional dependencies between functions Likely to transmit exchange items	*Activity Object Flow*	Defines the object flow between Activities and Actions Exchanges: Information Item

		By default they don't describe a sequence or chronology		
Differences		Differences of exchange items, see *Interface and data model*		
	Function	Is an action, operation or service by the system or one of ist components but also of actors interacting with the system	*Activity* *CallBehaviorActions* *Action*	Activity is a higher-level function CallBehaviorAction references an activity Action is an atomic function
Differences		An exchange item is created when a function is performed Differentiation between system or actor funtion (system function = green, actor function = blue)		No differentiation between system or actor function but between abstraction levels
	Function Port	Where the function interacts with other functions of its environment Input port: Requires specific exchange items	*Activity Parameter Node* *Object Nodes* *Pins*	Describe input and output on activity level Object nodes at the beginning and end of the flows that are used to accept inputs and provide outputs from activities Input and Output on action level
		See also Interface and data model		
	Not in Arcadia		*System Process*	Specify the logical order of execution of the System Use Cases Use of the system on a higher level than the System Use Case Setup of the system via some operational functions
	Function Scenario	Time-ordered dynamic flow on a temporal axis Set of functions and the exchanges that link them on a temproal axis System behavior in a particular usage context to contribute to one or more system capabilities	*Sequence Diagram*	Concrete collaboration of actors and/or system parts to perform a system function How parts of the system collaborate to perform a single path through a Use Case Activity
Differences		Used in Functional Analysis Exchanges between functions		Used in the Architecture Process Collaboration between actors and the system to perform functions

Structural Concepts

Arcadia		SYSMOD	
Actor	An actor is an entity that is external to the system (human or not) and interacts with it, especially via its interfaces	*User* *Environmental Effect* *External System (Mechanical system, Sensor, …)*	External entities that interact with the system of interest. SYSMOD provides several stereotypes to differentiate between actors
System	The system is an ordered set of elements functioning as a whole, responding to customer and user demand and needs, and subject to engineering supported by Arcadia Can have different specializations applying to different levels of abstraction - Logical Architecture - Physical Architecture Can be defined using diagrams but is not represented by a model element	*System & Subsystem*	Represents the system Subsystems for a breakdown of the system Can have different specializations applying to different levels of abstraction - Base Architecture - Logical Architecture
		System Context	Identify all users and other external entities that interact with the system Defines the environment of the system that needs to be considered, the system boundary, and the interfaces of the system.
Ports			
Arcadia uses the concept of Functional, Behavioral and Physical Ports and Exchanges. Capella allows modeling all three Ports on the System Analysis and Architecture levels. For the Functional Port see the Functional Concepts comparison. The Behavioral and Physical Ports are primarily used on the Architecture levels. That is why they will not be described in detail. Still it is important to show the main differences and similarities to the SysML Port concepts.			
Similarities	Arcadia and SYSMOD using SysML have built-in Ports. They both describe interaction points and exchanges/ flows between		
Differences	The Arcadia Ports have a built-in abstraction mechanism from high-level functional information to detailed exchange information (e.g., how communication between components is enabled). It is important to state that the Functional Ports are not part of the structural but the functional description		The SysML Ports and Flows (Full and Proxy Port) in general have no bult-in abstraction mechanism. But SYSMOD provides the Logical and Product Architecture where Ports can be abstracted and detailed
Behavioral Exchange Scenario	A behavioral component scenario is the time-ordered dynamic flow, on a temporal axis (conventionally vertical from top to bottom), of exchanges between different behavioral components in the context of implementing a capability.	*Sequence Diagram*	Concrete collaboration of actors and/or system parts to perform a system function. It shows how parts of the system collaborate to perform a single path through a Use Case Activity

Interface and Data model	
Arcadia	**SYSMOD**
Arcadia and SYSMOD/ SysML use similar concepts to describe functional and behavioral exchange and object and item flow.	
Arcadia uses Classes to describe data by defining the properties and value types of the data. The Classes can be grouped into Exchange Items. An Exchange item is used to describe the flow of functional and behavioral exchanges. An Interface can group Exchange Items that are semantically coherent. So, Arcadia uses a hierarchy to define what is exchanged between ports.	SysML uses the Item Flow to describe exchanges between ports owned by parts. The item flow defines source and target and the item that flows defined by a block. The block describes the structure of the item, e.g. by value properties. An Interface block specifies the I/O of ports by flow properties. SysML Object Flow describes exchanges between actions. But it is not expressed by an additional item flow. It defines the functional I/O by the pin types.
	Domain Knowledge: SYSMOD adds the concept of Domain Knowledge to specify the data, physical entities, related value types and units that are used by the system. They are represented by the block stereotype *<<DomainBlock>>* and can be used to model the item or object flow.