Team Solution Design White Paper: “System Context” Modelling

Ian Charters
Former IBM Distinguished Engineer (retired)

Version 1.1.1
04th January 2011

Table of Contents

Table of Figures

Introduction to Version 1.1 (10th December 2010)

In addition to numerous minor improvements and clarifications to the text, the paper has been brought in line with the latest release of Team Solution Design and the current configuration of IBM Rational System Architect for Team Solution Design (SA4TeamSD, version 1.14). In particular:

1. Discussions on the distinction between “internal workers” and “external users” has been removed from the body of the paper, with an expanded discussion on the topic included in “Appendix B: Modelling a System of Systems”.
2. Allied to this, “Appendix A: Users and Actors” has been greatly expanded, with a wider discussion on the need to understand the broader context in which the target IT system sits.
3. All SA4TeamSD screenshots and associated descriptions have been updated to be in line with SA4TeamSD V1.14.

Version 1.1.1 (4th January 2011) contains several minor corrections and a revision to Appendix C: Borders and Boundaries, including an additional figure.

Introduction

System context modelling provides an overview of the requirements placed on a target IT system within its environmental context, enabling the identification and description of its responsibilities to, and expectations of, other IT systems and human users (collectively referred to here as external agents).

Deciding on what constitutes "the target" and therefore "its environment" (i.e. everything else that the target IT system interacts with) depends on the circumstances surrounding the modelling efforts. For example, system context models can provide powerful insights when there is a need to

- Factor complex IT systems into smaller parts (whether exploring the structure of a complete enterprise or that of a major programme of change),
- Identify and agree to the span of control of a project team, which needs to confirm the nature of what's "in scope" for their project's responsibilities to and obligations from those users and systems beyond their direct control (and therefore "out of scope").

Typically, system context modelling takes two forms

- **Freeform system context diagramming** (or "sketching"), in which there is no or limited formal modelling notation or constraints on the nature and form of the diagram's content.

  In this case, the nature of the diagram's parts is left to the author to declare (such as is the case with the Architecture Overview diagram). This freeform approach enables a general description of the relationships between the target IT system, its parts[1], and its environment, albeit allowing for a high degree of ambiguity in the meaning of the diagram and therefore relying on guided interpretation.

- **Formal System Context Modelling,** in which a range of system context model views are drawn according to formally defined viewpoints, each intended to remove ambiguity of meaning and to provide explicit linkages with other models of the target system such as use case modelling, component modelling and operational modelling.

  In this form, the target IT system context's views explicitly document:
  - the target IT System as a bounded "black box", in which no internal functional or operational detail is described or illustrated,
  - the functional way in which the target IT system is used by (or uses) external agents, in both logical and physical terms
  - the operational manner in which these agents connect to the target IT system, in both logical and physical terms

The following section describes in detail the nature of the system context as a freeform sketch - referred to here as the target IT **System Context <u>Diagram</u>[2] (SCD)**; while the

[1] Strictly speaking, best practice system context modelling portrays the target IT system as a single "black box", although it is common to find it shown as a few interconnected parts – discussed later.
[2] In this paper, a <u>diagram</u> need not reflect the true underlying model nor be from a defined viewpoint.

subsequent sections focus on the formal mechanisms of system context modelling rendered as target IT **System Context Views**[3] **(SCVs)**.

In addition, each section includes a discussion on the way in which system context modelling has been implemented in the Team Solution Design (TeamSD) configuration of Rational System Architect, referred to as "SA4TeamSD":

- Team Solution Design is IBM's corporately agreed method for the design of IBM's IT based solutions, used by all Business Units in all Geographies. It is based on a suite of Artefacts defined in the Unified Method Framework, focused on the IT Architecture domain.
- In order to facilitate delivery of these solutions, the configuration of Rational System Architect to support TeamSD addresses all these artefacts at levels of detail appropriate for pre-sales as well enabling much deeper analysis and modelling for IT Architects working in post sales solution delivery.

The main example used throughout this paper is the IT Architect Profession's education case study "AmGro-from-Home", described in detail in section "*Example Target IT System: "AmGro-from-Home"*". All system context diagrams and views associated with this AmGro example are screenshots from the AmGro sample encyclopaedia (model) shipped with the SA4TeamSD configuration (version 1.14[4]) of IBM Rational System Architect (version 11.3.1, build 170), which are expected to be compatible with all subsequent releases of both SA4TeamSD and System Architect.

It is assumed that the reader is familiar with IBM's System Description Standard's metamodel and its key engineering perspectives [Ref 1], together with the Unified Method framework's use of architecture viewpoints & views [Ref 2].

Freeform System Context Diagramming

The IT System Context Diagram is a free form sketch intended to convey a general sense of the target IT system's context and responsibilities drawn in manners suitable for any number of different stakeholders. A project may therefore require many SCDs, each intended to support a specific purpose for a different specific audience, possibly at different but particular moments in time (and therefore seen as a specific "snap shot" of the target system's context).

The freeform nature of SCDs means they are most valuable early in the lifecycle (before or after formal project initiation) and for non-technical audiences, with whom their sketch like qualities make them a particularly powerful means of exposing and clarifying the requirements being placed on, as well as the expectations of, the target system. However SCDs are often not maintained, have ambiguous notation and may be in conflict either with each other or with design decisions recorded in "later" work products such as use case, component, or operational models[5].

---

[3] In this paper, a view reflects the true underlying model when observed from a defined viewpoint.
[4] Version 1.0 of this paper was compatible with V1.12.0.5.
[5] This does not necessarily reduce their value. If these characteristics are consciously acknowledged and accommodated then, for example, a time stamped or signed off SCD may still act as a record of agreement for a critical requirement or design decision between technical and non-technical parties.

A typical SCD portrays the context of one target IT System with any number of external agents, each of which may have one or more relationships with the target system, as illustrated in the following two examples.

SCD from UMF Artefact "System Context" (ART 0651)

At its simplest, the SCD provides no direct information on the nature of the external agent other than may be deduced by its name. In this most basic case, it is possible to assume some external agents are IT systems and others are humans and that, on the balance of probability, most if not all of its content is presented from the physical perspective:

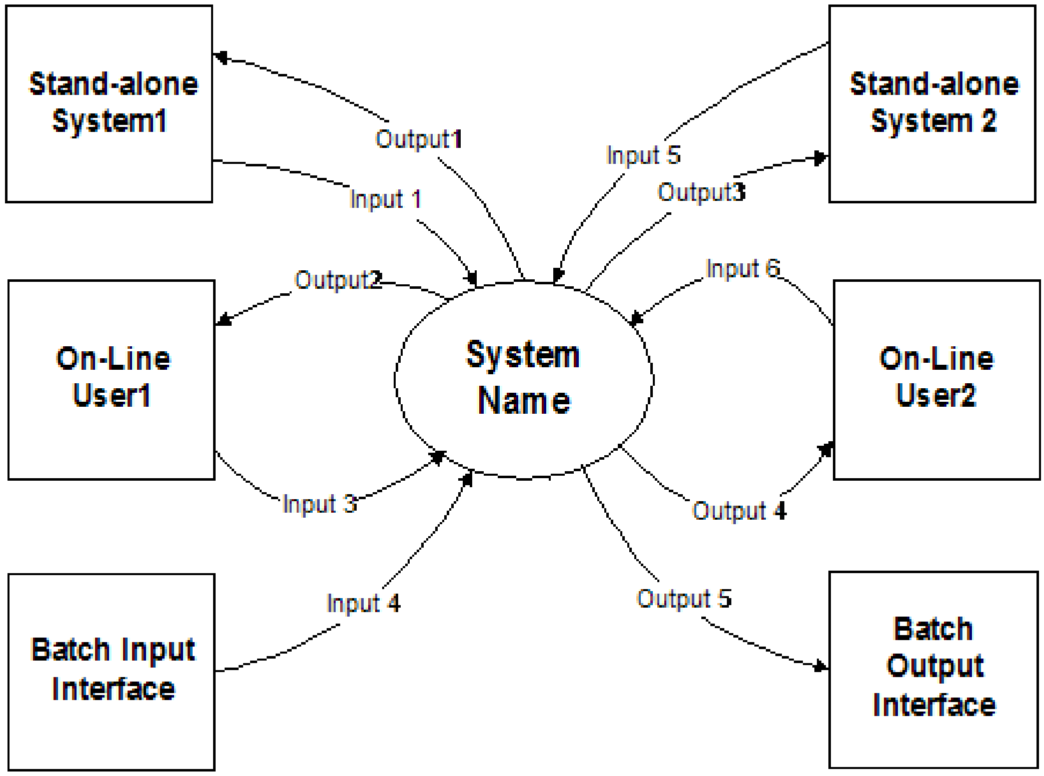


Figure 1: Standard System Context Diagram (example from UMF Artefact)

SCDs for "AmGro-from-Home"

This SCD, as illustrated in figure 2, is loosely based on the home shopping order management system described in section "Example Target IT System: "AmGro-from-Home""

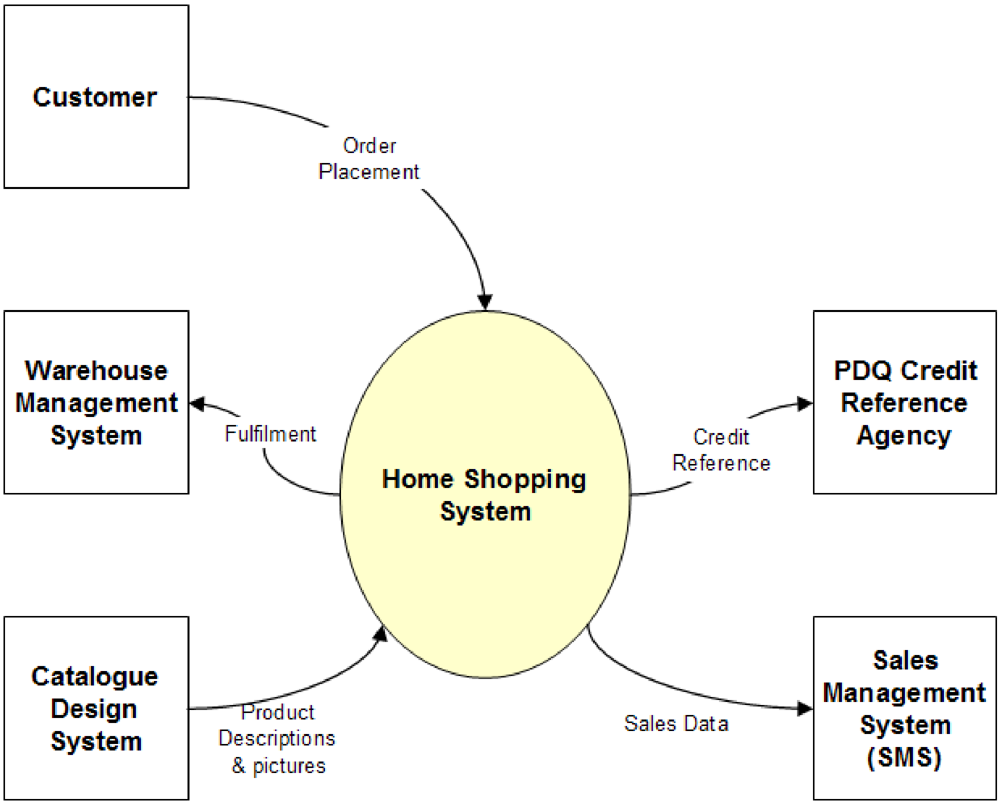


Figure 2: Standard System Context Diagram (simplified "AmGro-from-Home")

It also does not distinguish "human" from "IT System" - for example it's not possible to judge whether "customer" is a human using the Home Shopping System or a customer's PC; and neither can we be sure the "PDQ[6] Credit Reference Agency" is an on-line IT system, provides batch credit reference data, or even a human agent answering the telephone. Further, while words such as "customer" and the reference to "PDQ" may indicate a physical dimension to the diagram, "Warehouse Management", "Catalogue Design" and "Sales Management System" are all probably logical descriptions of several physical external IT systems (and which could even be one physical system being used in multiple ways!).

Additionally, and while not considered "best practice" by some, it can be helpful to use the SCD as an illustration of the target IT system as a few discrete (and interconnected) elements, such as may be needed to illustrate the scope of a project's sub-projects. It is also common to highlight relationships between the target IT system's external agents that are thought to "illuminate" the target system's context.

For example, Figure 3 highlights the decision to split the target IT system into two sub-systems; and how the target IT system is involved in a workflow closed loop, as well as being indirectly used by a set of external agents who also use it directly:

[6] PDQ = Process Data Quickly. PDQ is a common term for credit card terminals. This is a mythical company name

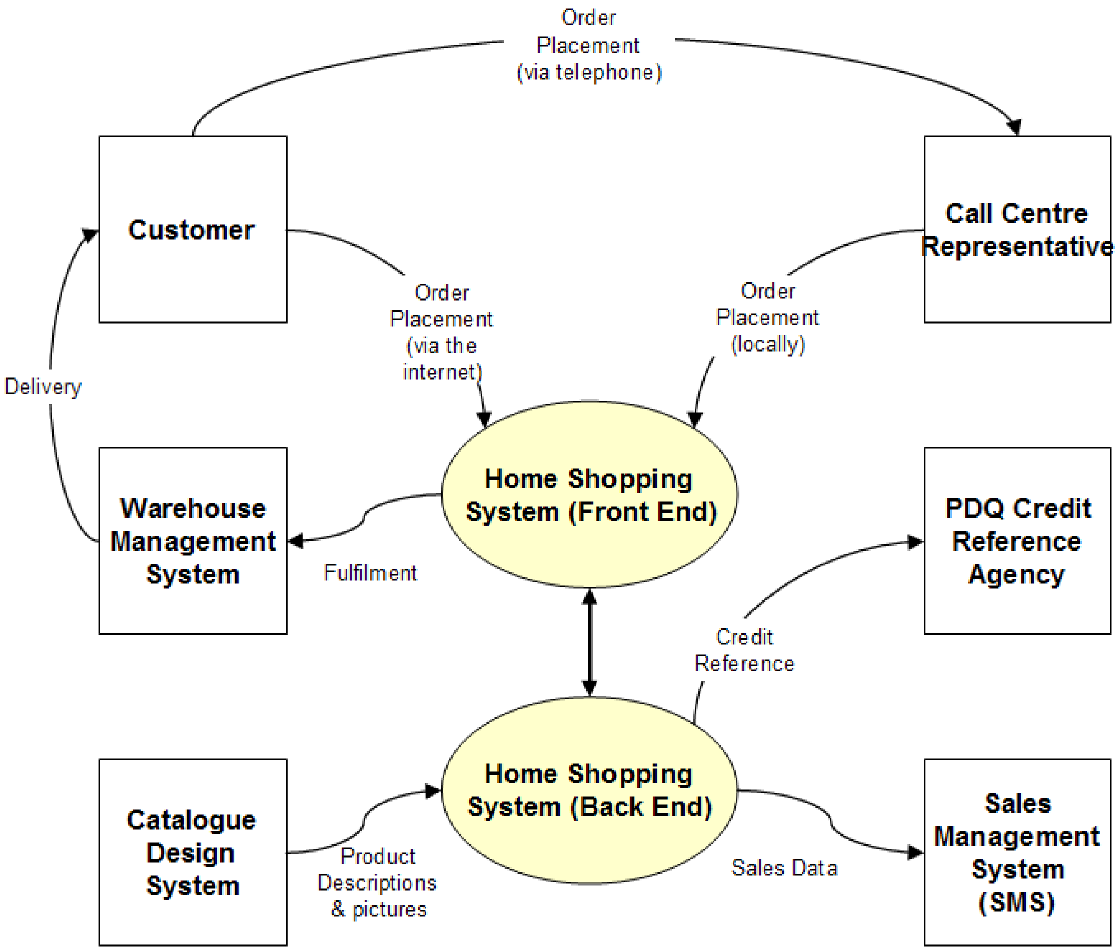


Figure 3: Extended System Context Diagram (simplified "AmGro-from-Home")

These additional elements on the SCD may make it difficult to distinguish it from an Architecture Overview diagram. However, while the AOD is completely freeform, an SCD is always restricted to show "the target IT system" itself at the heart of the diagram, the intention being to highlight its scope (albeit rendered in one or more parts) through descriptions of the relationships between it and its "external agents".

As these examples show, being a freeform diagram means an SCD has little or no formality in the precise meaning of the things that appear on it, specifically

- External agents can be of any form or type. Any one diagram can mix logical and physical agents, or may even choose to leave them indeterminate. Additionally, the agents may be declared to be human or IT, or again indeterminate.

- Relationships are described textually. Any one diagram may describe the functional way the system is being used by (or using) these agents, as well as the operational manner in which the agent is connected, or both, or neither.

- There is no formal notation such as the use of icons or arrow heads and other visual clues, such as may be used to indicate if it is the target system or external agent who initiates the relationship. All this is left to the writer – and therefore each reader's interpretation.

Interpreting an SCD (so that, for example, it can be assessed for completeness or consistency) can be aided by the provision of a key if a consistent (but private) set of icons and line types has been used. None the less, it is important to recall that the diagram is informal and that its content is always likely to be ambiguous. It also means that without the more formal views of a System Context model, identification of critical information may not happen until later tasks and work products and thus not be available for preliminary architecture discussions. Yet the SCD is helpful in basic enumeration of external agents and easily capturing known and constrained information.

System Context Diagramming and SA4TeamSD

An SCD can be created in the SA4TeamSD configuration as a type of "freeform diagram".

SA4TeamSD's freeform diagrams enable the modeller to create sketches illustrating any combination of model elements drawn from a pre-defined set of model element types appropriate to the specific type of freeform diagram[7]. In the case of SCDs these permitted model element types are:

- Target IT systems via the **Target System** model element type
- Logical external agents via the **Actor** model element type
    - Stereotyped as 'Human' or 'IT system'
- Physical external agents, via the **User** model element type
    - Stereotyped as 'Human' or 'IT System'

And each of which have the following standard SA4TeamSD icons:

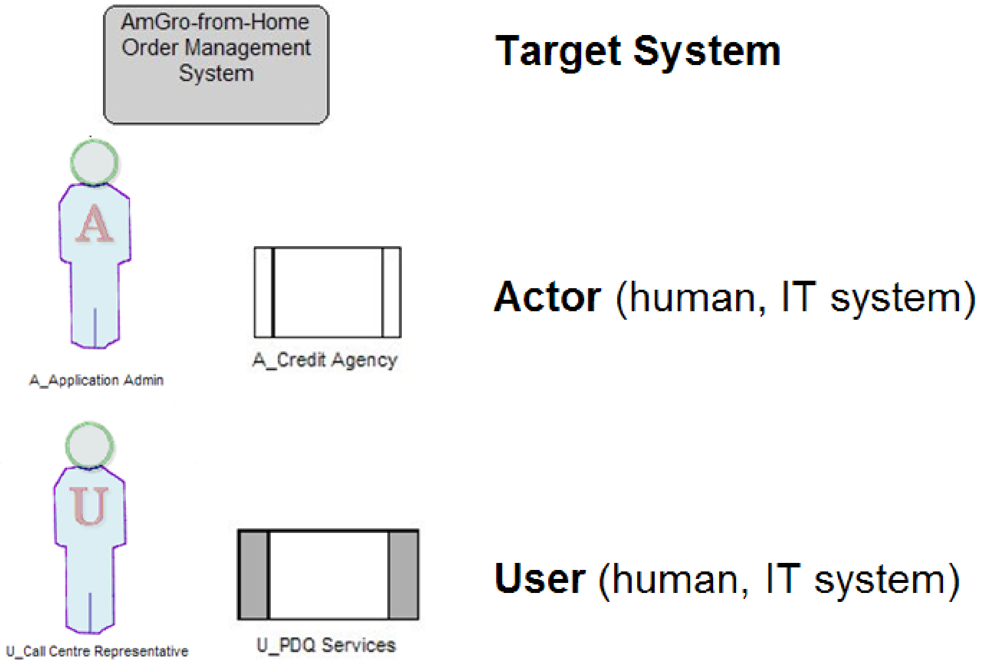


Figure 4: Standard SA4TeamSD icons[8] available for System Context Diagrams

For example, this figure presents the SA4TeamSD version of *Figure 1: Standard System Context Diagram (example from UMF Artefact):*

[7] Other freeform diagrams in SA4TeamSD include the Architecture Overview and Technical Environment.
[8] Logical external agents have "open" heads or sidebars, physical external agents are "filled in".

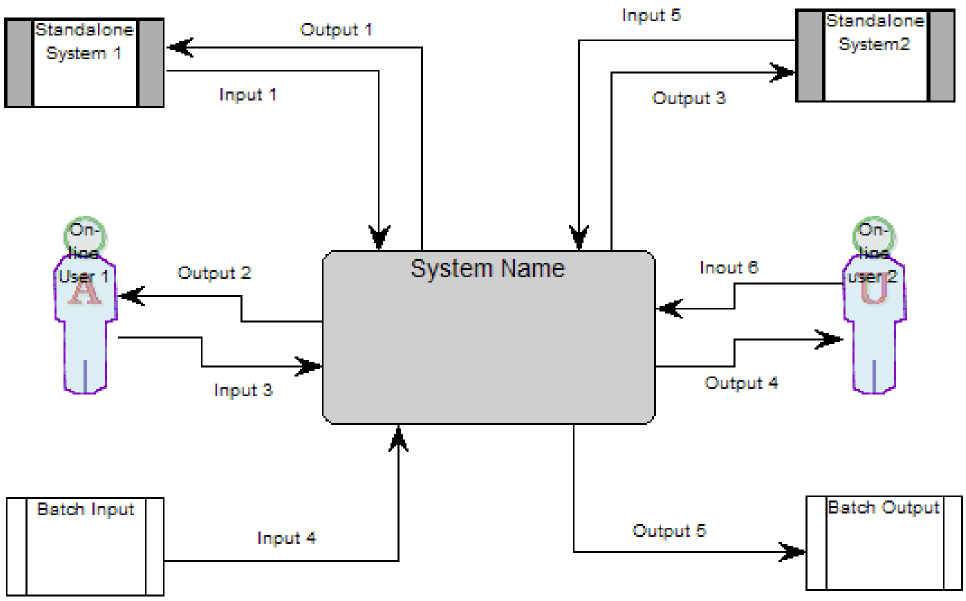


Figure 5: SA4TeamSD Standard System Context Diagram (example from UMF artefact)

In this case, SA4TeamSD's notation tells us that the

- "Standalone System 1" and "2" external agents are physical IT Systems
- "On-line User 1" is a human actor, whereas "On-line User 2" is a human user
- "Batch Input" and "Batch Output" are logical IT Systems

As with all SA4TeamSD freeform diagrams, the relationship between any two model elements (such as "Batch Input" and the target IT system in Figure 5) is documented as a "Freeform Connector" symbol[9], and therefore does not represent a relationship in the underlying System Architect model. They do, however, allow the modeller to portray otherwise contradictory or maybe "invalid" model element relationships, such as those between the target IT system's sub-systems or between external actors as shown below in the SA4TeamSD version of *Figure 3: Extended System Context Diagram (simplified* "AmGro-from-Home")

[9] SCDs do not portray "exchanges" or "access mechanisms" – these form part of the formal system context views discussed in the next section. However, the freeform connector's symbol's "graphic comment" field allows the diagram to suggest the nature of these relationships.

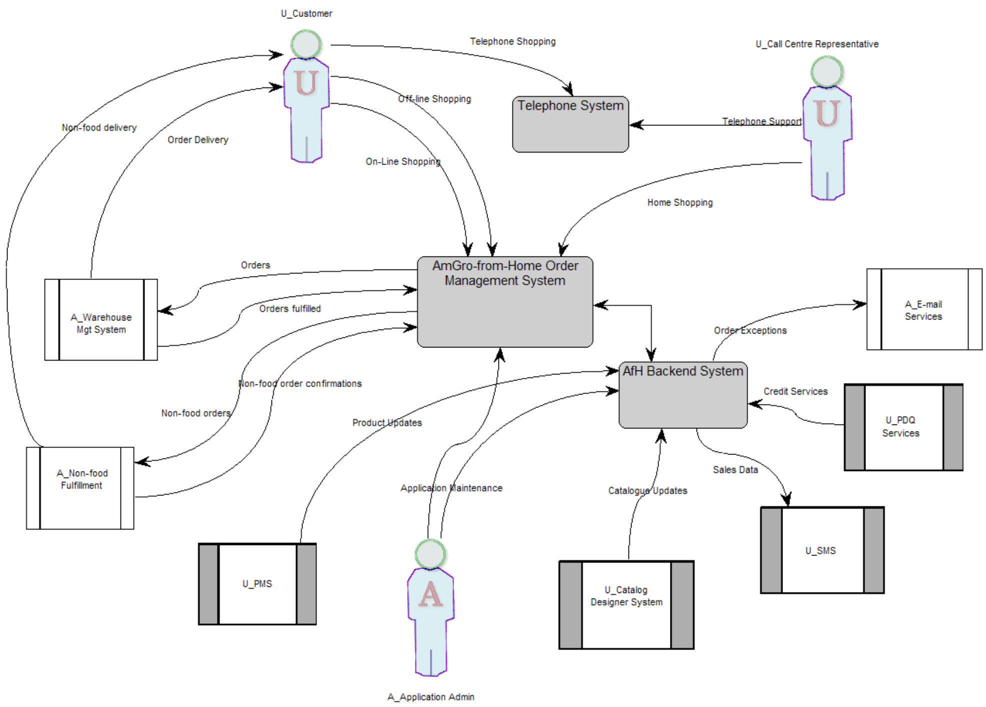


Figure 6: SA4TeamSD extended System Context Diagram (Full "AmGro-from-Home")

As is the case for all of SA4TeamSD's freeform diagrams, these two examples highlight how it is not possible to model "indeterminate" model elements on these diagrams: in this case it's not possible to model external agents that are intentionally indeterminate between logical and physical, or have not yet been determined to be a human or an IT system.

As a counterpoint to this, exploiting SA4TeamSD's SCD's blend of formality (model elements with standard icons understood by all users of the tool) and informality (permitting any and all relationships) supports the common practice of using freeform sketches to confirm the exact nature of the system's parts (such as distinguishing logical descriptions of requirement (actors) from physical constraints imposed by the nature of its human and IT system users), while allowing the modeller to gloss over the complexities of ambiguous or "invalid" relationships so that they can engage in "natural language" discussions and clarifications of the target IT system's requirements and obligations with "non-IT" audiences.

Formal System Context Modelling

The formal system context model's views provide unambiguous "black-box" statements about the logical and physical contexts in which the target IT system functions and operates, therefore being an important guide for the internal "white-box" systems design activities embodied in component modelling and operational modelling[10].

This is accomplished using a number of viewpoints (based on the formally defined engineering perspectives presented in [Ref 1]) from which the IT system's context is observed, with correspondingly formal views created, managed and used to communicate the system's contextual requirements:

- **Functional viewpoints**
  in which the system's functional responsibilities to, and expectations of, its external agents are analysed and documented. These viewpoints use the notion of **exchange** to describe what's going on between the external agent and the target IT system and are particularly useful in **component modelling**, whether this activity takes place after or in parallel with system context modelling. Two functional viewpoints are described:

  - Logical Functional viewpoint, describing the target IT system's "role based" relationships (**logical exchanges**) with its logical external agents (**actors**, whether 'human' or 'IT')

  - Physical Functional viewpoint, describing the target IT system's "real people and systems" relationships (**physical exchanges**) with its physical external agents (**users**, whether 'human' or 'IT')

  Exchange is discussed in detail in section "Modelling the System's Functional Boundary – "Exchange""

- **Operational viewpoints**
  in which the system's operational connectivity to its external agents is analysed and documented. These viewpoints use the notion of **access mechanism** to describe how external agents connect to the target IT system and are particularly useful in **operational modelling**, whether this activity takes place after or in parallel with system context modelling. Two operational viewpoints are described:

  - Logical Operational viewpoint, describing the nature of the "role based" connections (**logical access mechanisms**) between the target IT system and its logical external agents (**actors**, human or IT)

  - Physical Operational viewpoint, describing the actual connections (**physical access mechanisms**) needed between the target IT system and its physical

[10] This linkage between the black-box system context model and the white-box component and operational models is discussed in the component and operational modelling white papers [Ref 3 and Ref 4].

external agents (**users**, human or IT)

Access Mechanism is discussed in detail in section "Modelling the System's Operational Boundary – "Access Mechanism"

Representing these four perspectives visually provides an important insight into their relationships.

The Three Dimensions of Architecture Thinking

Figure 7 shows these four viewpoints as the plan view of the 2 by 2 by 2 "cube" occasionally used in other places; in which the 3rd dimension (i.e. the elevation views) highlights a 3rd engineering perspective:

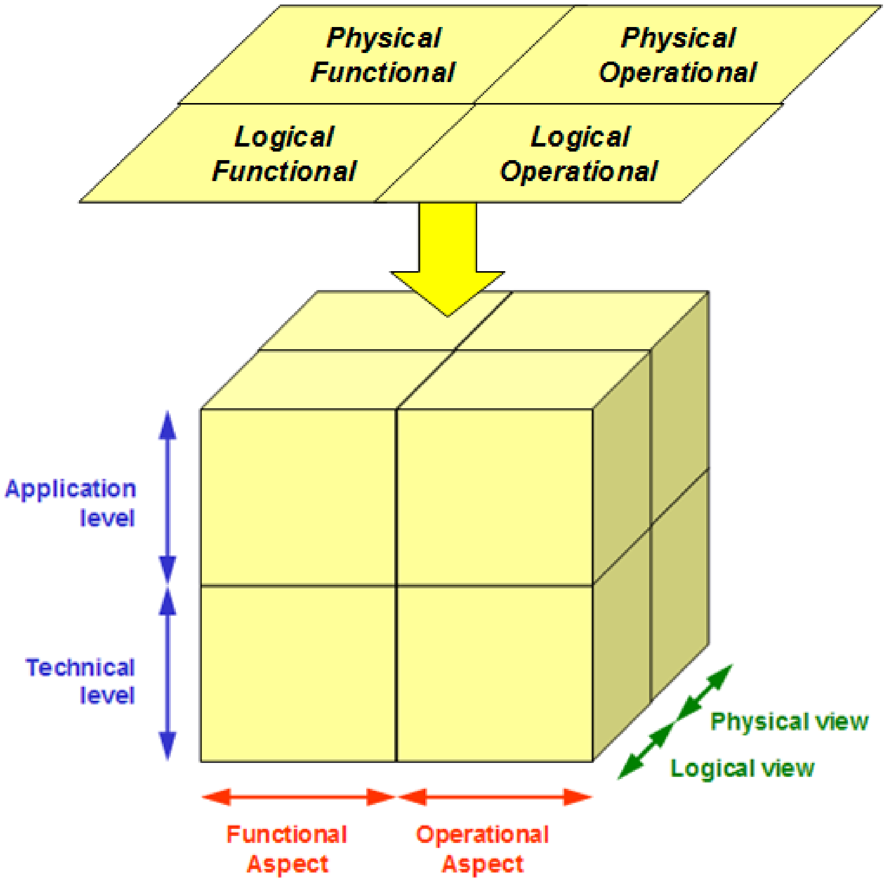


Figure 7: Positioning the 4 System Context viewpoints with the 3 dimensions of Architecture Thinking

These two levels distinguish the purpose of the parts being modelled:

- **The Application Level**, modelling those parts of the IT system associated with its target purpose and therefore the prime the focus of the modeller

- **The Technical Level**, modelling those parts needed to keep the IT system working and therefore usually "below" the prime focus of the modeller, whatever the primary purpose may be.

It may sometimes be necessary to exercise care when referring to these level names, since they are intended to distinguish relative levels of focus, not absolute levels; for example.

- In the AmGro example, those parts of the "AmGro-from-Home" order management IT system associated with the business purpose ("home shopping") is at the application level – which some may refer to as the IT system's "business level"[11].

- If, however, the target IT System was that of an enterprise's IT infrastructure, then those parts associated with it's main purpose ("application services") would still be modelled at the application level – which some may refer to as the IT system's "infrastructure level"[12].

As hinted by the way Figure 7 has been drawn "looking onto" the application level, it is typical for system context modelling to focus at the application level, particularly when exploring the requirements being placed on the target IT system. Subsequently, as decisions are taken to exploit external services rather than support them within the target IT system, it may be important to "look through" the cube into the technical level, altering the System Context views accordingly. This distinction has a particularly significant impact on the operational dimension - both for system context modelling (and the way access mechanisms are used) and operational modelling (and the use of presentation deployment units). This is discussed in detail in section "Modelling the System's Operational Boundary – "Access Mechanism""

(At this time, there are no notational means of distinguishing application and technical level content in system context modelling.)

Exploring the influence of these four viewpoints on System Context modelling and its use of the exchange and access mechanism notions depends on a reasonable understanding of the target IT system being modelled – in this case, the "AmGro-from-Home" order management IT system.

Example Target IT System: "AmGro-from-Home"

This section describes the way in which the "AmGro-from-Home" order management IT system is used (and uses) external agents[13]. It intentionally discusses the devices used by these external agents without concern for the target system's boundaries – in other words, whether the external agents' access devices are "inside" or "outside" the control of AmGro-from-Home.

As a consequence, the example avoids reference to the modelling mechanisms of "exchange" and "access mechanism" which are expressly discussed later in sections "Modelling System Boundaries – exchanges & access mechanisms". The example is presented as a simple, "sponsor friendly" architecture overview diagram:

[11] In some approaches to architecture, such as TOGAF and IBM's EA Method, this part of the IT domain is referred to as the Information Systems, or IS level.
[12] In some approaches to architecture, such as TOGAF and IBM's EA Method, this part of the IT domain is referred to as the Technology level.
[13] The example has been carefully constructed in order to provide the necessary content needed to discuss system context modelling

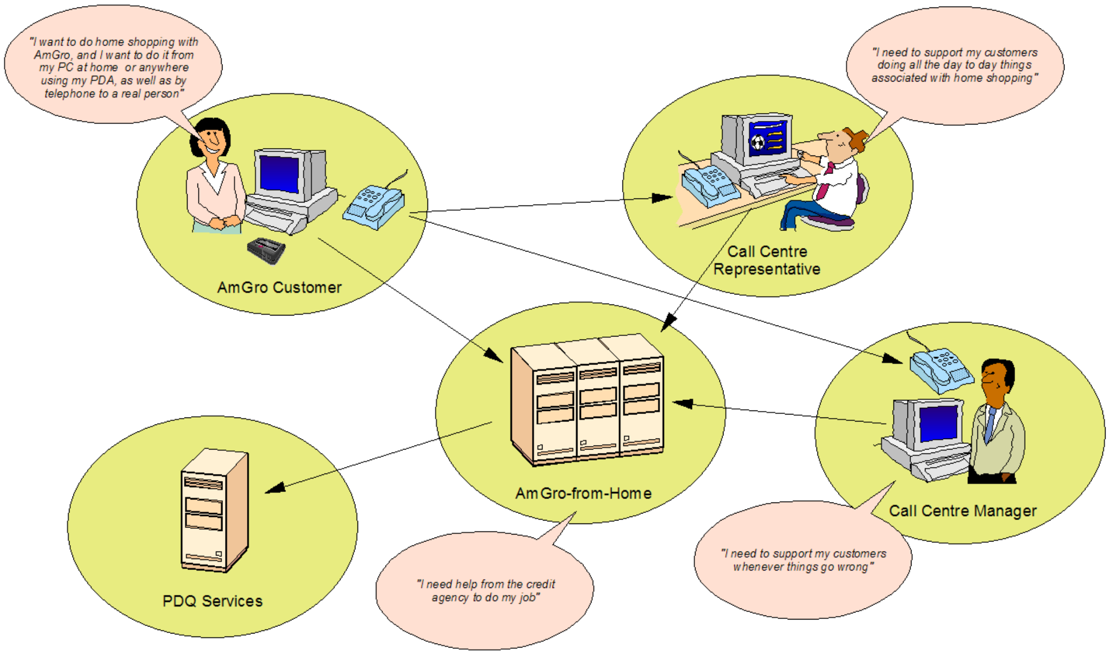


Figure 8: AmGro-from-Home Architecture Overview Diagram

Figure 8 shows how three separate users of the home shopping system all want to do the same sort of thing, accessing the system in a variety of ways and from various places, while the home shopping system itself uses an external agent for credit card pre-authorisations. All this has been documented in figure 9 as a freeform system context diagram (drawn in SA4TeamSD), in which the different ways used to access the system by AmGro's customer is explicitly documented by showing the customer's PC (or PDA) as an additional external agent:

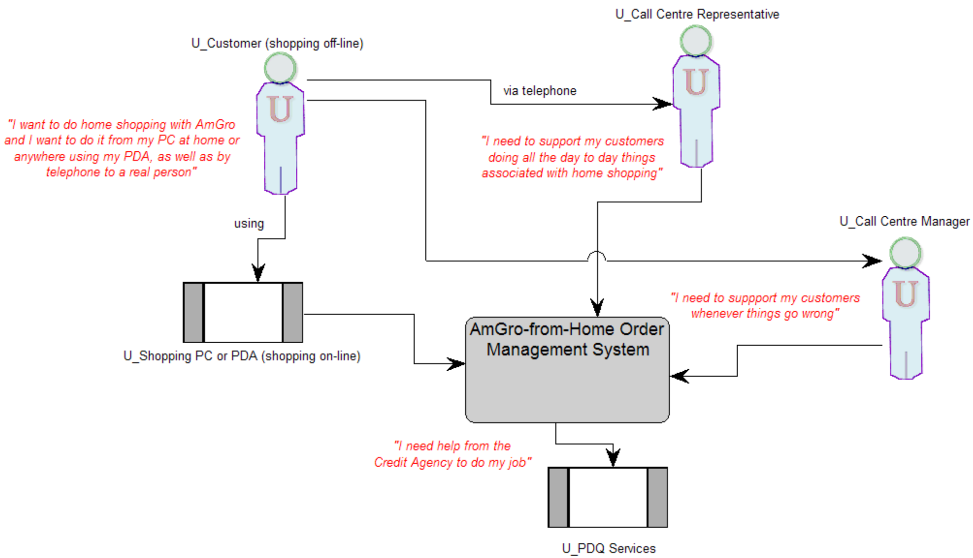


Figure 9: Freeform System Context Diagram (AmGro-from-Home)

(It may be observed, from the comments in the section on "Formal System Context Modelling" that Figure 9 illustrates both application level and technical level contexts, insofar as the three users are associated with the system's business purpose, while a technical decision has been taken to use an external agent for credit card pre-authorisation.)

More formally, the next four sections analyse and document the system and its external agents using two of architecture's three dimensions discussed in section "Formal System Context Modelling":

- Viewpoints across the functional dimension, focusing on the people using the system (the physical users) and what they are doing (as logical actors)
- Viewpoints across the physical dimension, looking at how these users physically access the system
- Viewpoints across the operational dimension, exploring how the users' activities are influenced by the logical capabilities of their access mechanisms
- Viewpoint across the logical dimension, establishing what the users (as actors) functionally want to do via these logical access capabilities

This analysis uses the same standard model element naming conventions used in the AmGro encyclopaedia, in which the name of a model element is prefixed according to its type:

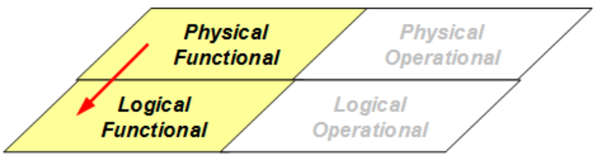
- U_ for model elements of type "user"
- A_ " " " "actor"
- PN_ " " " "physical node"
- LN_ " " " "logical node"

Viewpoints across the functional dimension

The 'at-home' AmGro Customer, Call Centre Representative and Call Centre Manager each represent different "real people" and can therefore be modelled as three different (physical) users within the context of the home shopping system's IT system – put differently, the 'Customer' user group, 'Customer Service Representative' ('CSR') user group and 'Call Centre Manager' user group each represent the sum of all the things done to (or for) each group of people by the system[14]:

- **U_Customer**: the 1000's of people wanting to buy from AmGro whether by telephone or PC / PDA
- **U_Call Centre Representative** (U_CSR): the 10's of AmGro employees taking and managing orders by telephone. These users are able to support all aspects of home shopping, except the handling of customer complaints which must be referred to a CSR's manager.
- **U_Call Centre Manager**: the 1's of AmGro managers handling customer complaints
- **U_PDQ Services**: "PDQ Ltd." is the credit reference agency used by AmGro

From the earlier picture we can deduce that each of these physical users (groups of users) are functionally doing slightly different home shopping things, each of which can be modelled as different (logical) home shopping roles - in other words mapping physical users to specific logical actors doing specific functional things with the target system:



[14] This is an appropriate simplification. See "Appendix A: Users and Actors" for a deeper discussion.

Actor (role of a User)	U_Customer (1000s)	U_CSR (10s)	U_Call Centre Manager (<10)	U_PDQ Services (1)
A_Home Shopper	Yes	Yes		
A_Complaining Shopper	**?**		Yes	
A_Credit Reference Service				Yes

The table shows how the human users engage with the target system in similar roles, i.e. as actors[15] - although in different combinations (illustrating the many-to-many relationship between ‘actor’ and ‘user’). These roles can be described as:

- **A_Home Shopper** represents “an ordinary shopper”; the day-to-day” role most customers perform - either directly with the system (via their PCs and PDAs) or via telephone, in which case the role is performed on the target IT system “by proxy” by CSRs.
- **A_Complaining Shopper** describes the role of someone who wants to complain about the home shopping service. The “?” in the table is a critical aspect of the example – we do not yet know if this role can be performed directly by customers (i.e. when they are on-line) but know that, at a minimum, it is available to them via telephone via Call Centre Managers (not CSRs) who use the target IT system on their behalf.
- **A_Credit Reference Service** represents the role of the PDQ service’s system in its relationship with the home shopping system.

The terms “by proxy” and “on their behalf” warrant further discussion. As was illustrated in the system’s freeform SCD, home shopping via telephone is *in scope* of the overall Information System (a.k.a. Business) system. However, the telephone call handling system supports the customer’s call to the CSR, who in turn is using the same target IT system as the customers are using when they are on-line and therefore customer access via telephone is *outside the scope* of our target IT system. This distinction between the contexts of Business and IT systems is briefly discussed in “Appendix B: Modelling a System of Systems”, highlighting how the CSR might be modelled as a “worker” (that is. inside the Business system, but outside the IT system) rather than an “actor” (which would be outside both the Business and IT systems).

This functionally oriented “black box” analysis is most commonly applied in use case and component modelling. In these cases, given that the focus is on “application functionality”, uncertainty about who will act in which role and how they will access the system can be tolerated: for example whether it’s the customer or the Call Centre manager who is complaining, or whether they’re complaining via the telephone or on line could be irrelevant – “customer complaints” code is needed either way.

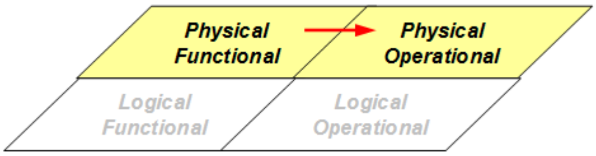
[15] This notion of actor being a role is based on the “Actor” artefact from the Unified Method Framework.

So to help ensure the system is deployable and operable, this functional dimension should be complemented by an operationally focused analysis, again embracing both the logical and physical perspectives.

Circumstances may indicate which of these perspectives should be tackled first – for example the ambiguity on how customers can complain may indicate priority is given to the logical dimension. But in this instance, we've chosen to first "look across" the square on at the physical level (in the next section) before navigating "downwards" through the operational dimension (in the subsequent section).

Viewpoints across the physical dimension

The physical viewpoint explores the manner in which the target IT system's various (physical) users physically access the home shopping system:



In this case, the freeform SCD confirms that there are three (physical) devices being used by the human users of the home shopping system, together with one for the external IT System:

Access Device (physical node) used by each user	U_Customer	U_CSR	U_Call Centre Manager	U_PDQ Services
PN_At home PC	Yes			
PN_PDA	Yes			
PN_Call Centre PC		Yes	Yes	
PN_PDQ interface PC				Yes

In which:

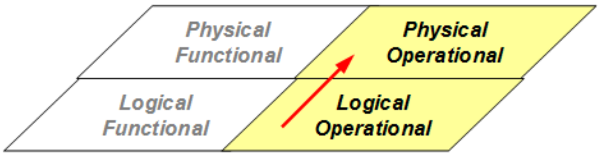
- **PN_At home PC** is the personal (privately owned) PC used by AmGro's customers
- **PN_PDA** is the personal digital assistant (such as a Blackberry® or similar device) privately owned by AmGro's customers
- **PN_Call Centre PC** is the AmGro PC being used by AmGro's employees in the AmGro call centre.
- **PN_PDQ interface PC** is the PC supplied by PDQ and installed by them on-site at AmGro to ensure the link back to PDQ is guaranteed secure.

(Note how this example substantiates the many-to-many relationship between users and access devices – one user using two devices and two users using the same device.)

The following analysis illustrates how knowing what users want to logically do functionally and how they want to physically connect operationally to the IT system (as was also shown in the typical SCD presented earlier) often does not provide sufficient information for the system's design.

Viewpoints across the operational dimension

Further requirements analysis across the operational dimension may highlight what the (physical) user can or wants to do (logically) is often influenced by the way they want or can exploit the (logical) capabilities of their (physical) access device[16].



In our example, we may discover that

- AmGro's customers want (for reasons such as cost, performance or privacy) to do some parts of their home shopping logically "off-line" from the home shopping service, which ever device they're on
- AmGro's employees' workstations may be constrained (for reasons of security or availability) to be permanently connected to the home shopping system whatever they do
- The PDQ service may operate in two logical modes, with credit card pre-authorisation sometimes being performed locally, and on other occasions requiring greater scrutiny back on the external PDQ system.

Access Device (physical node) being used as (logical node)	LN_off-line (Customer's Private device)	LN_on-line (Customer's Private device)	LN_CSR w/station (AmGro's own device)	LN_Credit Gateway (PDQ's device "on site")
PN_At home PC	Yes	Yes		
PN_PDA	Yes	Yes		
PN_Call Centre PC			Yes	
PN_PDQ Interface PC				Yes

In which:

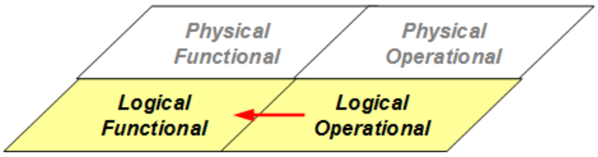
- **LN_off-line** describes the logical capability of any private physical device that's being used for home shopping in a "disconnected mode" from the central AmGro home shopping system
- **LN_on-line** describes the logical capability of any private physical device when it's connected remotely to the AmGro-from-Home order management IT system
- **LN_CSR w/station** describes the logical capability of an AmGro PC when it's connected locally to the AmGro-from-Home order management IT system
- **LN_Credit Gateway** describes the logical capabilities of the PDQ PC, whether satisfying the credit card pre-authorisation request directly or relaying it onto PDQ head office.

(Note how this example substantiates the many-to-many relationship between physical devices and the manner in which they are used – one physical device is being used for several purposes, and different physical devices are being used for the same purpose[17].)

[16] This will probably requires further negotiation of the business requirements!

Viewpoints across the logical dimension

Critically, the analysis so far still does not tell us what the users want or can do (logically, as actors) under each of these different (logical) operational ways of using the (physical) access devices – can the customer complain on line?



After extensive analysis and further requirements negotiation, it turns out that for many practical operational reasons,

- it is only possible to enable AmGro's customers to do a very limited amount of "home shopping" off-line, whether using their PC or PDA: they are to be restricted to managing their shopping lists and some aspects of their personal profiles. Shopping itself will only be possible when they (or rather their access device) is connected to the home shopping system[18]
- Data management limitations between the order management and warehouse management systems (Figure 3) mean it will not be possible to allow customers to complain on-line. They will therefore have to be restricted to the telephone, where experienced Call Centre managers will be able to handle their complaints "by proxy".
- AmGro's CSRs and their managers can act in all roles using the same logical access device (which we learnt previously was implemented one way)
- All of PDQ's credit reference service can be supported through the one gateway

Actor according to logical access device (logical node)	LN_off-line (Customer's Private device)	LN_on-line (Customer's Private device)	LN_CSR w/station (AmGro's own device)	LN_Credit Gateway (PDQ's device "on-site")
A_Shopping List Manager	Yes	Yes	Yes	
A_Home Shopper		Yes	Yes	
A_Complaining Shopper			Yes	
A_Credit Ref Service				Yes

Comparing this revised set of actors (and their descriptions) with those described earlier in section Viewpoints across the functional dimension" - particularly the insight that "A_Complaining Shopper" cannot be done using the customer's own private devices highlights how the operational viewpoints influence the specification of a system's requirements, and that a purely functional analysis may fail to discover vital information:

- **A_Shopping List Manager** *(new)* represents "an ordinary shopper" who's preparing and maintaining their shopping lists.

[17] albeit probably with different interface characteristics. This distinction is discovered, analysed and documented using the Presentation Deployment Unit mechanism in Operational Modelling.
[18] It may be enlightening to learn that in the early days of grocery home shopping; widespread practice was to provide a full function shopping experience "off-line", using file transfer mechanisms to update off-line PC based catalogues and to submit completed orders.

- **A_Home Shopper** *(modified)* represents "an ordinary shopper" in almost all aspects of the home shopping role except for those associated with shopping list management and complaints
- **A_Complaining Shopper** *(unchanged)* describes the role of someone who connects to the target IT system with a complaint about the home shopping service.
- **A_Credit Reference Service** *(unchanged, clarified)* represents the role of PDQ Service's system, whether performing "real time" credit card pre-authorisations, or providing (via daily file transfer) a "hot card list" for transactions not requiring pre-authorisation.

(Note how this example substantiates the many-to-many relationship between actors and their logical devices – one logical device is being used by several different actors, and two actors are each using different logical devices.)

Modelling System Boundaries – exchanges & access mechanisms

The example above describes and documents the AmGro-from-Home order management IT system by a direct analysis of who its external agents are (as "actors" or "users"), together with a description of the devices they use (logical and physical).

As such, it

- Overlooks the distinction between a user's role and what they are doing to the target IT system in that role
- Has no concern for whether they (or their access device) is inside the target IT system (and therefore its responsibility), or outside (and therefore not).

The next two sections discuss how these functional and operational "boundary conditions" are more accurately modelled as exchanges (in the System Context Model's Functional Viewpoints) and access mechanisms (in its Operational Viewpoints), first in terms of the notions themselves, and then by way of the Amgro-from-Home example.

Modelling the System's Functional Boundary – "Exchange"

The notion of exchange enables the model to explicitly separate what someone or something is described to be (as an actor or user) from what someone/something does with a particular system (described as their/its exchange with that system):

- *An exchange aggregates all the interactions across all the collaborations in which the external agent participates.*

Exchanges and the "AmGro-from-Home" order management system

For example, the following diagram models a minimum part of the Amgro-from-Home order management IT system's logical functional context:

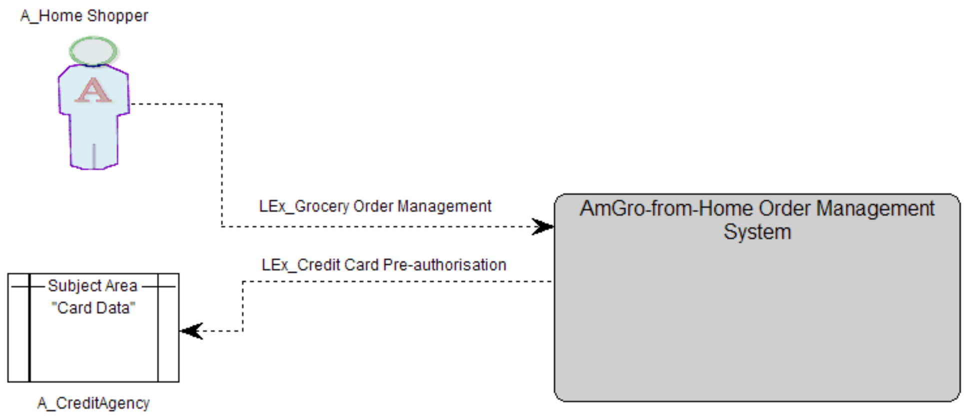


Figure 10: Actors and Logical Exchanges (AmGro-from-Home)

In which:

- Actor **"A_Home Shopper"** is a description of the responsibilities and expectations of all people who wish to "home shop", whatever system they may be using (AmGro-from-Home or not), together with a description of their characteristics such as their level of expertise (both in using the home shopping application as well as IT in general), cultural background and so on[19], together with how many of them may be using the AmGro-from-Home order management IT system.

- Exchange **"LEx_Grocery Order Management"** is a description of all the interactions between the actor A_Home Shopper and their use of this specific "AmGro-from-Home" order management IT system, whatever the channel they may use: as we saw earlier, this is everything except shopping list management and complaints, both of which can only be performed on a subset of access mechanisms.

- Actor **"A_Credit Agency"** is a description of the responsibilities and expectations of the IT system(s) being used to provide credit reference services, together with a description of its characteristics such as responsiveness, hours of availability, transaction cost and so on[20].

- Exchange **"LEx_Credit Card Pre-authorisation"** is a description of all the interactions that take place between the actor A_Credit Agency and this "AmGro-from-Home" order management IT system, including file transfer (hot card lists) or real-time transactions (credit card pre-authorisation).

As is also shown on Figure 10 an exchange may be modelled as directional[21], influenced by the predominant nature of the exchange's various triggers – they may be inbound from the actor (as shown by the arrow head at the inner end of LEx_Grocery Order Management); outbound from the target system ("LEx_Credit Card Pre-authorisation") or both.

This example illustrates the one-to-one relationship between actor (in this case A_Home Shopper) and logical exchange (LEx_Grocery Order Management) *for any one target system.* This can lead to a similarity of names when modelling within a single context, which on the

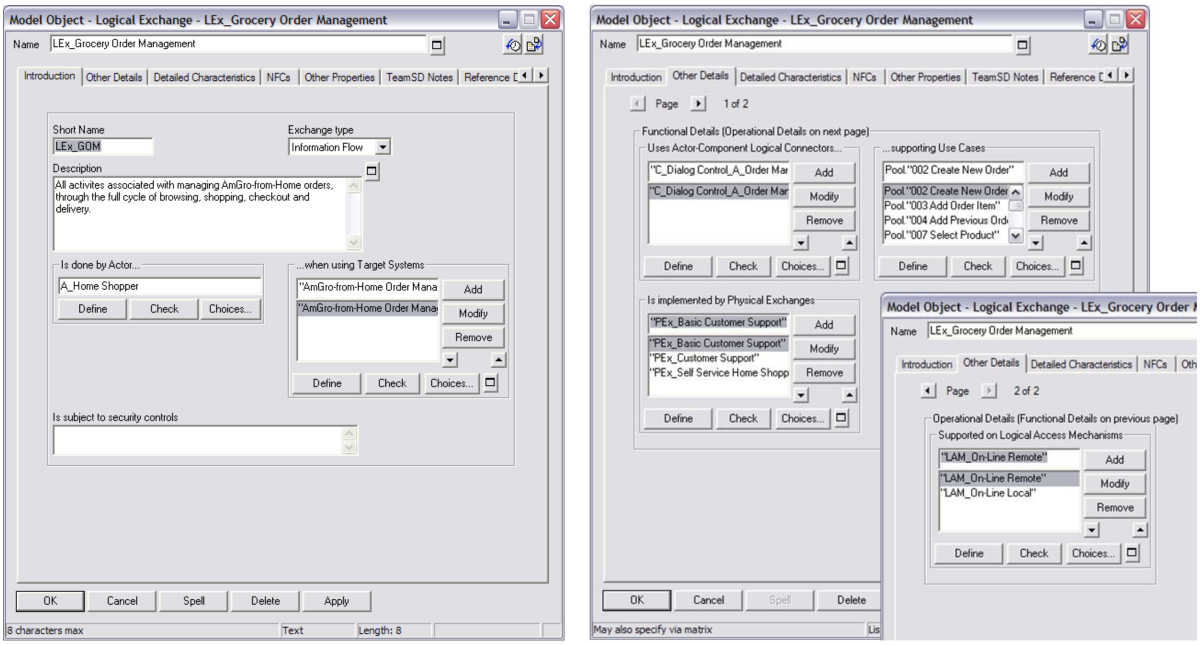
[19] The notion of actor as a role rather than a set of activities is discussed in the UMF "Actor" artefact.
[20] The "Actor" artifact is less explicit on the nature of IT system actors.
[21] Using the freeform connector symbol's "associative" property.

face of it suggests either "actor" or "exchange" is a redundant modelling notion. But blurring what someone (or something) does to or for a target system with a description of who they are, what they know or how they act is not good practice[22].

Thus the phrase "*for any one target system*" is vitally important: an external agent (whether actor or user) may use many different IT systems, not just this target system, for all of which they are (must be) described the same way. But in each case they will have one "exchange" to describe what's going on between them and each specific system they are using. The value of this separation is therefore much more obvious when the target system is being reasoned about in a broader context (such as in a programme or enterprise), when the same actor (i.e. any number of different users all in the same role) is doing a different thing to or for each separate system. Here, for example, each of the AmGro-from-Home actors may be using (or be used by) other AmGro IT Systems, such as if AmGro wants to:

- Think about the **A_Home Shopper** role in a consistent manner no matter how they go about shopping with AmGro (in store or on-line)[23], or maybe in an industry standard way for market management purposes.
- Use the **A_CSR** on a "Loyalty Scheme" help desk as well as the AmGro-from-Home shopping service. In this case, the same actor has different exchanges with each system.
- Use the **A_Credit Agency** for its existing in-store systems as well as their home shopping system. In this case, the actor has the same functional exchange with each system although probably with different non-functional characteristics

Additionally, SA4TeamSD's definition for logical exchange provides additional detail not visible from Figure 10:



[22] We do not confuse the idea of a component with a description of what it does for other components!
[23] Which may indicate a better name for the role would be "A_Shopper"?

Figure 11: Logical Exchange Definition (SA4TeamSD)

The definition provides a summary description of what a logical external agent (i.e. an actor) is doing to or for the target IT system, relating it to detailed descriptions of the requirements placed on the target system by the associated actor[24], such as

- The use cases in which the actor is involved, both as a primary or secondary actor.
- The set of functional requirements being placed on the target system by the associated actor.
- A description of the target system's state changes as triggered by (or affecting) the involved actor

It also describes how the exchange is enabled by the connector(s) between the target IT system's components and its associated actor and is supported via one or more operational logical access mechanisms (discussed in detail in the next section). This detail provides a cross-reference to the internal "white box" functional design of the target system, which may in turn influence the external "black-box" design: for example it may be that the nature of the access mechanism prevents certain functionality, or that several components are needed to support the actor, which may trigger a refactoring of the "black box" model.

Most importantly, the definition describes how the logical exchange is accomplished[25] by one or more (directional) physical exchanges, which can also be modelled diagrammatically as shown in Figure 12.

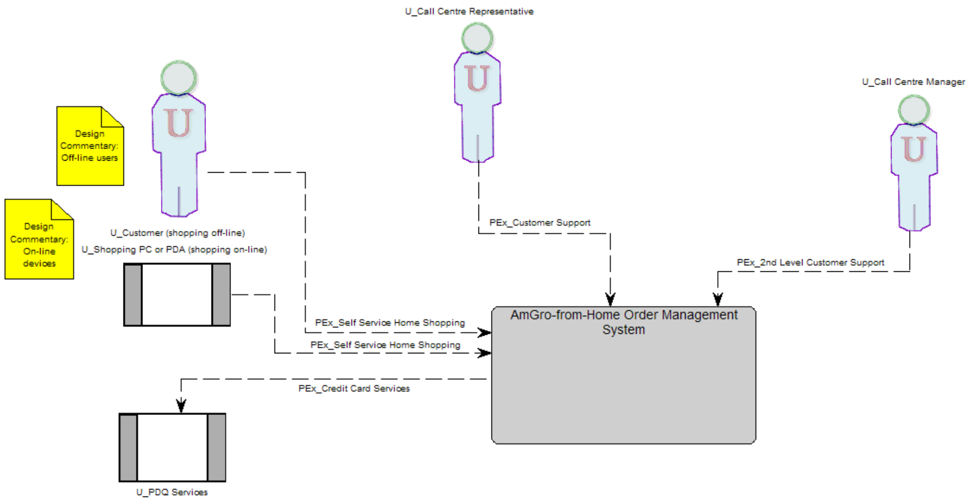


Figure 12: Users and Physical Exchanges (AmGro-from-Home)

A physical exchange summarises all or some of the interactions of one or more of the physical external agents (users) when using the target system (a many to many relationship), and may therefore implement one or more logical exchanges[26]. Figure 12 also highlights

---

[24] Systems design methods may use a variety of approaches to model the external requirements being placed on a Target System. For example, "functional requirements" is common means of requirements specification when focused on infrastructure design; and "state machine" modelling has advantages when dealing with certain forms of complex systems. SA4TeamSD supports Use Case modelling directly, and function requirements via the component associated with the exchange.

[25] Avoiding the ambiguous term "realised" ☺

[26] Since a logical exchange's actor may be implemented by many users, so a logical exchange may be implemented by many physical exchanges

(via yellow “TeamSD Notes”) the “cross-over” between the functional aspect’s exchange (what an agent is doing) and the operational aspect’s access mechanism (how and where agents connect to the target system: it shows two users associated with the same “PEx_Self Service Home Shopping”, in which one user is the human customer, and the other is their IT system (PC or PDA).

Modelling the System’s Operational Boundary – “Access Mechanism”

Locating and describing the target IT system’s boundary[27] is one of the core purposes of the system context model’s operational viewpoints.

However, in the earlier section “Example Target IT System: “AmGro-from-Home””, the operational aspects of the AmGro-from-Home order management IT system’s requirements were described and documented using an analysis of the external agents’ physical and logical IT devices (whether the agents were human or IT), and did not consider

- whether these devices were inside the target IT system (and therefore its responsibility) and leading to the system’s edge or boundary being with the external agent;
- or whether these devices were themselves also outside the target IT system and therefore behaving as the agent’s proxy with the target IT system (i.e. the device itself is the target IT system’s external agent).

Thus, our ultimate goal in understanding the target IT system’s operational boundary is to know:

- **<u>where</u> the system’s boundary is**, so that we can be sure of our target IT system’s scope - our boundary may be:
    - adjacent to the system’s external agents,
    - adjacent to the “edge” of the target system,
    - on an external agent.

- **the <u>nature</u> of the boundary**, between the system and its external agents, so that we can be sure we connect to our target IT system’s users in the right way - in particular (<u>at the system boundary</u>) to know if the system is interacting with:

    - Humans, and therefore needing to support a Human-Machine Interface (HMI) such as screen and keyboard,
    - IT systems and therefore using some form of communications protocol such as HTTP/HTML or file transfer.

This operational context – describing the location and nature of the system’s edge or boundary for each of the target IT system’s users, is accurately modelled via the notion of access mechanism:

- *An Access Mechanism describes the location and nature of the connection between the target IT system and one or more external agents, and is at the boundary between them.*

[27] “Boundary” has a specific meaning here - see “Appendix C: Borders and Boundaries”

Access mechanisms are similar to node-to-node connections, insofar as they describe the connection between the external agent and the target system (just as connections connect nodes). Therefore, like connections, they have two ends (one at the target IT system and one at the external agent). However, unlike most[28] connections, both ends of an access mechanism are in the same place (at the system's boundary), one end being the target IT system's edge, and the other being the edge of the external agent.

As a very simple example, consider the diagram in Figure 13, showing a target IT system connected to an external IT system via an Ethernet network, in which two alternative *in extremis* decisions on the target IT System's boundary (i.e. its operational scope) have been highlighted:

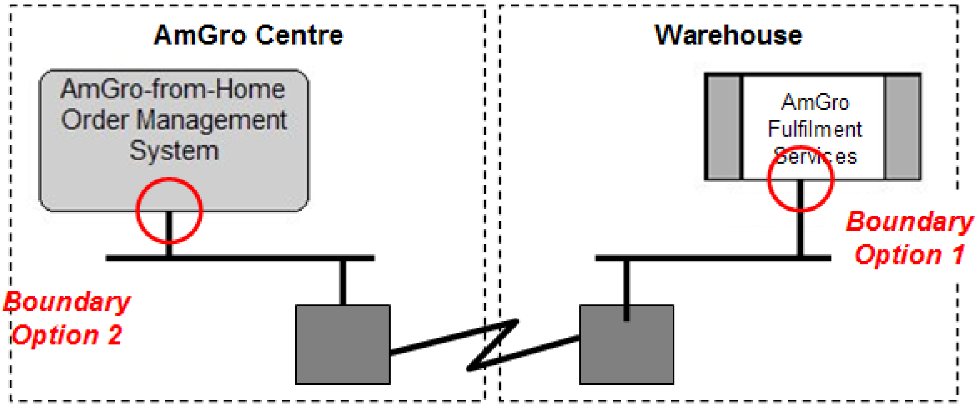


Figure 13: A simple system with two alternative boundaries...

The target IT system's boundary could be

- **Option 1:** in which the entire network is "in scope" for the target IT system. The access mechanism is therefore located in the external IT system's location and its nature will be described as the end of the Ethernet cable plugged into target IT system's Ethernet socket.
- **Option 2:** in which the entire network is "out of scope" for the target IT system. In this case the access mechanism is in the target IT system's location and its nature is described as the Ethernet socket "on the back" of one of the target IT system's nodes (or maybe the Ethernet fly lead connecting it to a local Ethernet hub).

These two options would be documented on two different (physical) operational views of the target IT system's context, showing two different access mechanisms:

[28] If two connected logical nodes are implemented on the same physical node, then the ends of their logical connection are technically in the same place.

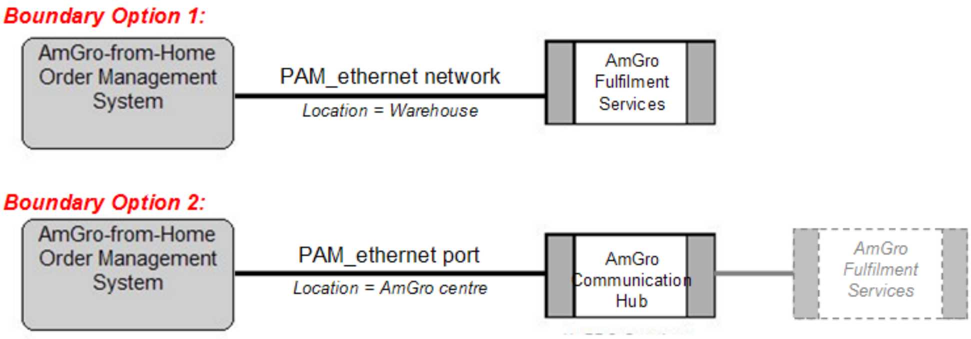


Figure 14: ...and the two corresponding operational system context views

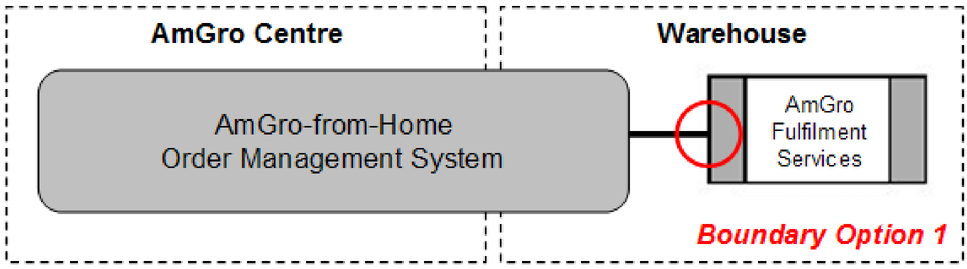
In the first option, the external system's access mechanism is described as being the target system's Ethernet network, available to the external system in its own location. But in the second, the access mechanism is described as an Ethernet port located with the target system[29].

As highlighted by the accuracy of Figure 14's system context views, the sketch in Figure 13 can now be seen to be technically incorrect in several ways:

- For option 1 the target IT system includes the network, whereas the picture shows the network separately from the target system. In other words, the target IT system actually extends into the external IT system's location

- For option 2 the strict modelling shown in the lower view highlights how, in physical terms, the external agent turns out not to be the external IT system, but a pre-existing communications hub co-located with the target IT system and acting as a "proxy" for the external IT system.

  The two options shown in the diagram on Figure 13 should therefore be drawn thus:



And

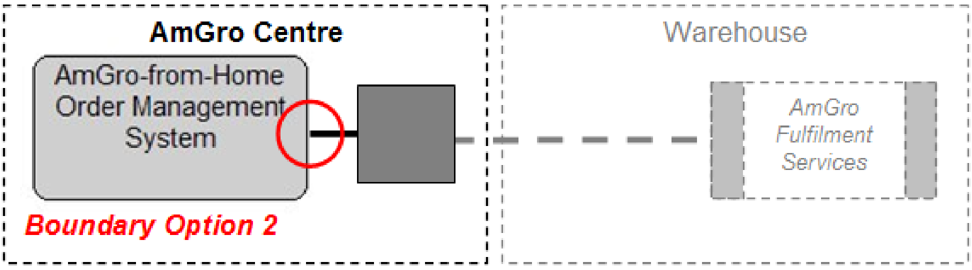


Figure 15: the same simple system drawn more accurately for each option

[29] Please wait two paragraphs for a discussion on option 2's two external agents!

It is probable that the first diagram in Figure 15 will emerge as part of the modelling activity. However the second one is less intuitive and may even be unhelpful. While it is undoubtedly accurate, there are numerous occasions when the view shown in option 2 will cause confusion – because it is too technical. Thus, and as was discussed in section “The Three Dimensions of Architecture Thinking”, it is more normal to model a target IT system’s context at the application level, in which the technical underpinnings (in this case the network) are omitted from the view. This ensures the external agent is modelled as the external IT system, although this then means the location of the access mechanism’s external agent is therefore different from that of the target IT system’s edge[30].

For this example, the application level operational view of the system’s context for option 2 is legitimately drawn thus, with the access mechanism’s (application level) external agent being the external IT system, and its location being recorded as “warehouse”:

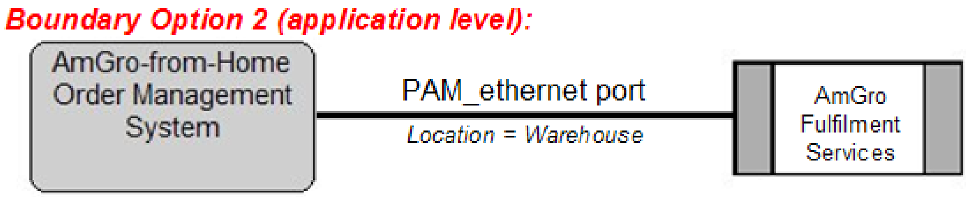


Figure 16: Option 2 in Figure 15, drawn at the application level

The next few pages show in their analysis of AmGro-from-Home order management system that identifying and describing the system’s boundary may require careful analysis and modelling in order to expose uncertainty in the requirements, not only when it is unclear whether the “external agent” is in the same place as the target IT system’s boundary (as discussed here), but also how it is necessary to carefully understand the nature of the external agent itself.

Access Mechanisms and the “AmGro-from-Home” order management system

Figure 17 illustrates the application level logical operational view of the “AmGro-from-Home” order management system’s System Context, showing the logical access mechanisms being used by the target IT system’s actors:

[30] It’s as if the exchange disappears through the target IT system’s edge, only to miraculously re-appear at the external IT systems edge: two sides in different places. Remind you of “Star Gate”®? And no, I’ve no idea how it works either. Here we are consciously choosing to model the same sort of behaviour.

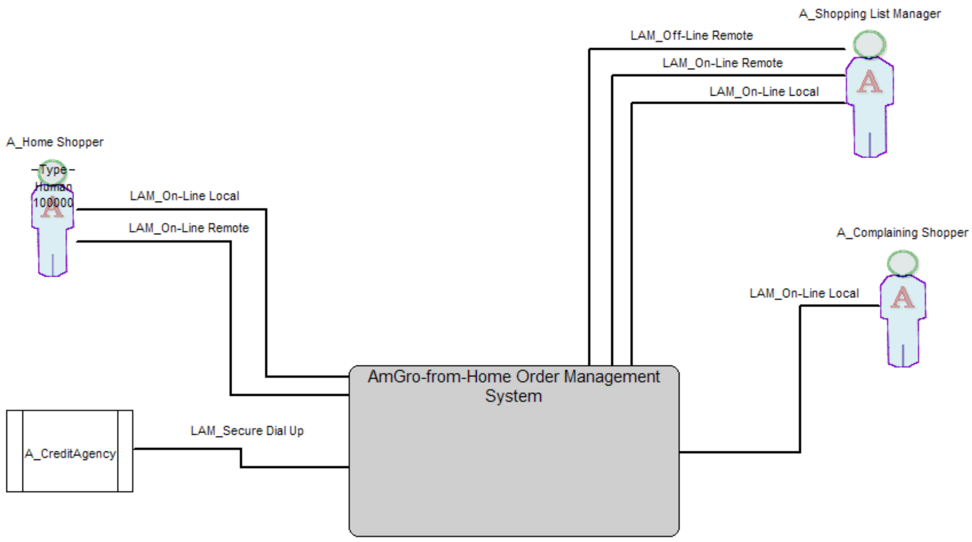


Figure 17: Actors and Logical Access Mechanisms (AmGro-from-Home)

It shows:

- Actors "**A_Shopping List Manager"**, **"A_Home Shopper"** and "**A_Complaining Shopper**" as before.

- These actors are using access mechanisms:
    - **"LAM_Off-line Remote"** that enables actors to use the home shopping system, but in a disconnected manner. This access mechanism is "anywhere" the actor wishes it to be.
    - **"LAM_On-line Remote"** that enables actors to access the central home shopping system's full range of capabilities anywhere they are able to connect to the internet and use a device enabled to support home shopping.
    - **"LAM_On-line Local"** that is provided to actors via PCs provided as part of the Amgro-from-Home target IT system

- Each of the system's actors are able to use one or more of these access mechanisms, depending on which user is in the role and where they are able to access the system from:

    - Users in the role of "A_Shopping List Manager" may access the AmGro-from-Home order management system in all three ways:
        - Remotely, off-line – these actors are AmGro's customers
        - Remotely, on-line – these actors are AmGro's customers
        - Locally on-line – when the actor is a role of AmGro's CSRs

    - Users in the role of "A_Home Shopper" may access the system in one of two ways
        - Remotely on-line – these actors are AmGro's customers
        - Locally on-line – when the actor is a role of AmGro's CSRs

    - Users in the role of "A_Complaining Shopper" can only access the system
        - Locally on line – we know these actors to be Call Centre Managers

- Actor **"A_Credit Agency"** as before

- Access Mechanism **"LAM_Secure Dial Up"** describing the manner in which the AmGro-from-Home order management IT system uses a dial-up mechanism to access the various services offered to it by the A_Credit Agency, whether these be to access hot card lists or real-time credit checking.

It also shows that access mechanism has no direction – unlike exchange, it is neither "in-bound" nor "out-bound" (nor both), and therefore access mechanisms cannot be drawn with arrow heads.

A number of key properties of logical access mechanisms are not visible in Figure 17, but are captured in the SA4TeamSD definition:

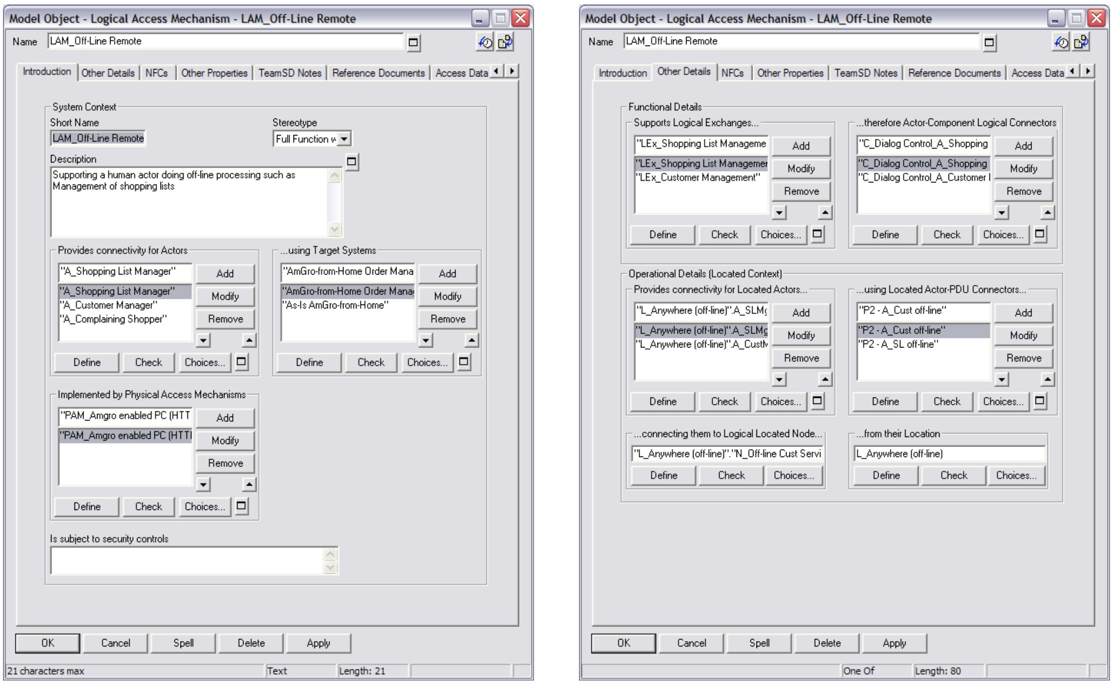


Figure 18: Logical Access Mechanism Definition (SA4TeamSD)

The "Introduction" tab of this definition documents, together with the actors who use this LAM to access various target IT systems:

- The physical access mechanisms (PAMs) that implement the LAM and which relate to the LAM's actors' users – which may be one or more for each LAM, as shown in Figure 19 below.

In addition, the "Other details" tab describes the LAM in more detail, including some functional aspects:

- The "black box" functional exchanges supported across the LAM – these should be associated with the actors using the access mechanism

- The "white-box" functional connectors responsible for the messages that flow between those actors and the target IT system's components.

As well as several operational aspects:

- The location of all a LAM's located actors ("*Provides connectivity for located actors…*" "*…from their location…*"), as well as identifying the system's located node to which the LAM is connected ("*…connecting them to logical located node*"…).

- It is also possible to model the connectors needed between these located actors and the presentational interfaces of the target IT systems' "edge components" placed on the LAM's located node ("*…using located actor – PDU connectors…*")

  In other words, a logical access mechanism enables a collection of actors to access the target IT system from a particular location[31], bridging them to the boundary of the target system (the location of the access mechanism's located node) and carrying their interactions with the target IT system's edge components.

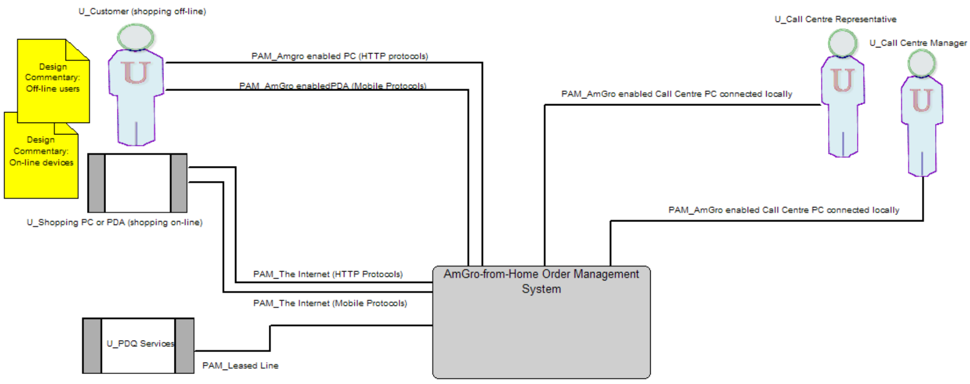


Figure 19: Users and Physical Access Mechanisms (AmGro-from-Home)

This physical view of the system context model's operational dimension illustrates how

- The call centre's users (CSRs and managers) connect to the AmGro-from-Home order management system using "**PAM_AmGro enabled Call Centre PC connected locally**". Strictly speaking, this access mechanism defines the machine-human interface of a PC, provisioned as part of the target IT system and locally attached to the main AmGro-from-Home order management system.

- The PDQ organisation uses "**PAM_Leased Line**", a leased line to access the target IT system, as the implementation chosen on this occasion for the logical requirement for a "dial up connection".

- AmGro's customers connect to the AmGro-from-Home order management system in four ways, depending on what they are doing and how they are doing it:

  - "off-line" for shopping list management, using AmGro code installed on either
    - Their own PC ("**PAM_AmGro Enabled PC**")

[31] To be clear – all the actors need to be in the same place to use the same LAM (the LAM's location); although an actor can access the system from many places (needing a LAM in each place).

  - Their own PDA (“**PAM_AmGro Enabled PDA**”)

- “on-line” (via their own device as their proxy) to AmGro’s internet service for shopping list management and home shopping, using either
  - Their own PC (“**PAM_the internet (HTTP protocols)**”)
  - Their own PDA (“**PAM_the internet (mobile protocols)**”)

Three types of access mechanism

Earlier, reference was made to three possible types of system boundary, that in turn leads to three types of access mechanism:

1. access mechanism for a human external agent
2. access mechanism for an IT System external agent
3. access mechanism on an external device being used to host an “edge” part of the target IT system

Each is exemplified in the AmGro-from-Home example, and can be summarised thus…

Type of access mechanism	**Sub type**	**Access Mechanism location, when modelling at**		**Access Mechanism nature, when modelling at**	
		Technical level	**Application level**	**Technical Level**	**Application Level**
Type 1: Connection to a Human	-	Local to the human external agent		Human interface	
Type 2: Connection to IT System	-	Local to the target IT System’s edge node	Local to the external IT system’s interface	Machine interface	
Type 3: Hosting part of target IT system on an external device and…	..connecting to an external IT System	Local to the external device	Local to the external IT system’s interface	Machine interface	
	..connecting with a Human	Local to the human external agent		External device interface	Human interface

The next three sections provide a detailed discussion, highlighting the variations in the access mechanism’s location and nature (human or IT), depending on the level of modelling (technical or application).

LAM type 1: connection to a human external agent

This type of LAM occurs whenever the device being provided to the target IT system’s human user is being deployed as part of the target IT System - the example’s **“LAM_On-Line Local”** access mechanism is of type 1.

This access mechanism is being used to support the target IT system's various logical exchanges with three actors – "**A_Shopping List Manager**", "**A_Home Shopper**" and "**A_Complaining Shopper**". These three roles are only performed by the AmGro-from-Home Call Centre's CSRs. Thus:

- The <u>nature</u> of the access mechanism is the human-machine interface (HMI) of the PCs supplied as part of the AmGro-from-Home order management system ("**PN_Call Centre PC**").

- The <u>location</u> of the system boundary, and therefore the location of this type of access mechanism is always adjacent to the human external agent whether the access mechanism is being modelled at the application or technical level.

  Thus in this case the application and technical location of "LAM_On-line Local" is the Call Centre (where the CSRs and managers work).

LAM type 2: connection to an IT system

This type of LAM can be found in two circumstances:

- <u>Directly</u>, when the target IS System's external agent is an IT System. Therefore the target IT system's external agent is also an IT System.

  This is exemplified by the example's "**LAM_Secure Dial up**" access mechanism, being used to support the target system's exchanges with the IT system "**A_Credit Agency**"

- <u>By Proxy</u>, when the target IS system's external agent is a human, but the device being used by them is already in place and the target IT system has no responsibility for it.

  This is illustrated via the example's "**LAM_On-Line Remote**" access mechanism. This is being used to support the logical exchanges associated with the actors "**A_Shopping List Manager**" and "**A_Home Shopper**" – roles only available via this access mechanism to AmGro's own customers.

The second example warrants further discussion. In "Example Target IT System: "AmGro-from-Home" it was described how AmGro's customers are required to use their own devices for on-line access to the central AmGro-from-Home order management IT system. Further analysis tells us that, on balance, it will be best to do this *without installing or running any part of the AmGro-from-Home order management system* on "**PN_At home PC**" or "**PN_PDA**". This means AmGro's customers' devices are external to the "AmGro-from-Home" target system and therefore

- The access mechanism's <u>nature</u> is always with an IT System, described in terms of the protocols and network connectivity needed to support the exchanges with the customers' devices.

- However, the <u>location</u> of the AM needs further analysis, and depends on the level of the system context's model:

- At the technical level, with the model focusing on all the underlying technical things needed to keep the system working, it considers itself to be interacting with something "immediately adjacent" such as a pre-existing AmGro router or hub. The technical level access mechanism would therefore be associated with the location of this device on this side of the internet, similar to the sketch of "boundary option 2" in Figure 15

- At the application level, with the model focusing on the business related aspects of the target IT system, it considers itself to be interacting with the external IT system. The application level access mechanism is therefore associated with the location of the customer's PC or PDA which is the other side of the internet, as was sketched in Figure 16.

The phrase "*without installing or running any part of the AmGro-from-Home order management system*" makes this LAM type distinctly different from LAM type 3, although both can be deployed using the same technologies, such as a web browser:

- Without local applets or cookies is an implementation of LAM type 2
- With local applets or cookies is an implementation of LAM type 3

LAM type 3: hosting part of the target IT system on an IT system external device

This type of LAM, in which *part of the target IT System is installed and runs on an otherwise external device,* is both the most complex as well as one of the most common. It exhibits two subtypes:

- **Type 3a: Hosting external device is a proxy for the external IT System.** For example:
  - Systems using a web access mechanism (such as a browser) that install local applications (applets) and data (cookies) onto their users' PCs, and thence communicate with these applets[32].

  - It may be necessary for a target IT systems to access an existing "backend" external IT system that was designed to support a different protocol to that being used by the target system. It is therefore necessary to install a "wrapper" on the external IT system in order to convert between the legacy and new protocols[33].

- **Type 3b: Hosting external device is a proxy for the human external agent**, Where the target IT System installs an application directly onto an end user device, "driving" the end user's experience directly from that application

  This is exemplified in the "AmGro-from-Home" order management system by the

[32] rather than the browser's interface, which would make it a LAM type 2 (assuming the browser to be pre-existing)
[33] This is not exemplified in the AmGro-from-Home example. However a good example is "screen scraping" software that provides access to external IT systems expecting to communicate with human users

"LAM_Off-Line Remote" access mechanism, where the human home shopper uses their own PC to run AmGro's own home shopping application to access some "off line" functionality of the AmGro-from-Home order management system.

In this second case, the access mechanism is being used to support the logical exchange "**LEx_Shopping List Management**" between the AmGro-from-Home order management system and the actor "**A_Shopping List Manager**" - a role only available via this access mechanism to AmGro's own customers.

Section "Example Target IT System: "AmGro-from-Home"" described how AmGro's customers would need to use their own PC (or PDA) to work off-line. Further analysis tells us that, in order for these devices to be used in isolation from the central home shopping system, it will be necessary to install elements of the AmGro-from-Home order management system onto the device "**LN_ off-line**" (whether this be their physical device "**PN_At home PC**" or "**PN_PDA**").

This means that the customer's PC (or PDA) is internal to the "AmGro-from-Home" order management IT System[34], and therefore

- The access mechanism's <u>location</u> is wherever the customer's own device (and therefore the customer) is located.

- However, the <u>nature</u> of the AM needs further analysis, and depends on the level of the system context's model:

    - At the <u>application level</u>, the model focuses on the business related aspects of the home shopping system and may therefore consider its "edge device" to be interacting with the human shopper. The access mechanism is therefore associated with the <u>human user</u> and describes the screen, keyboard, mouse etc of the customers' own device(s)[35].

    - At the <u>technical level</u>, the model must describe how the needs laid out at the application level are delivered, and therefore the access mechanism models the system's boundary being with the pre-existing device, describing the protocols used with the <u>external IT device</u> (such as HTTP/HTML for web type devices).

Access Mechanisms and "edge nodes" in Operational Modelling

When the modeller eventually moves from black-box to white-box modelling and looks inside the system (as part of operational modelling) they model the "inner end" of the access mechanism[36] as being attached to a node at the edge of the target IT system. The nature and location of this edge node varies according to the type of AM it is supporting:

---

[34] In Operational Modelling this hybrid between an external physical IT system and an internal logical node is referred to as an "External Node".
[35] This is similar to the "Presentation Deployment Unit" approach supported in operational modelling.
[36] Strictly speaking, an access mechanism does not have "an inner end" – an access mechanism has a property defining its location to be that of the located actors/users using it. However, it is connected to a located node, whose location is at the access mechanisms "other end".

- **AM Type 1 – with a human**
  - Implementation: the target IT system's logical edge node (ie the node responsible for the human's logical access mechanism) is always implemented as a physical node within the system
  - Location: At all levels of the Operational Model (application and technical) both logical and physical nodes are co-located with the human external agent[37].

- **AM Type 2 – with an IT system**
  - Implementation: the target IT system's logical edge node is always implemented as a physical node within the system
  - Location: For the application level of the OM, either/or/both logical and physical node may be local or remote from the IT system external agent. However, at the technical level, both must be co-located with the IT system external agent's interface.

- **AM Type 3 – hosted on an IT System**
  - Implementation: The target IT system's logical edge node is implemented on an external IT system, modelled as a "physical external node"
  - Location:
    - If the ultimate user is human, the logical edge node and physical external node must be co-located with the ultimate user
    - If the ultimate user is IT, they may or may not be co-located with it.

All of these variations are exemplified and discussed in [Ref 4].

Actors: Human or IT System icons?

On all forms of system context diagrams and views it is common practice to use different icons for human actors and users (such as an AmGro customer and their various roles) from those used to represent IT systems (such as the Credit Agency). Unfortunately, doing this in a consistent fashion is not always possible on the system context views.

For example the actor **"A_Home Shopper"** has been modelled with two access mechanisms – "**LAM_On-Line Local**" (LAM type 1) & "**LAM_On-Line Remote**" (LAM type 2):

- In the type 1 case, the "A_Home Shopper" is a role of AmGro's CSR, using a PC that is part of the target system – therefore the access mechanism is to a human, and the LAM's actor's icon should be "human".

- In the type 2 case, AmGro's customer is using their PC or PDA to access the home shopping system, which is therefore the target IT systems external agent "A_Home Shopper". Therefore in this case the LAM's actor's icon should be that of an IT system.

[37] We assume than humans have to be co-located with the device they are using to access the IT system!

The “A_Shopping List Manager” actor’s three LAMs exemplify all three types or actor (human, IT system and external node) since the actor’s use of “LAM_Off-Line Remote” is a type 3 LAM, being the customer’s own device hosting AmGro code.

Appendices

Appendix A: Users and Actors

Strictly speaking (and as defined in the "Actor" artefact of the Unified Method Framework), the notion of *a user of an IT system is defined to be a single user* – either a single person or a single external IT system. As such, it would be possible for each and every target IT system to exhaustively describe each user in fine detail, in terms of what they are doing and the characteristics they have in doing it, thereby distinguishing them from all other users.

In system modelling such as that discussed here however, it is generally more useful to exploit a degree of abstraction, modelling users as sets of groups, in which all members of the group are "doing the same thing"[38] with the target IT system. Each of these functional groups of users describes *a specific role of a collection of individual users, and is referred to as an Actor.* Thus for any one target system there can be a many-to-many relationship between "user" and "actor" – any given user can act in many roles, and any role (or actor) can be accomplished by a number of different users.

For most systems, however, the modelling of single users rapidly becomes unwieldy, particularly if there are large numbers of very similar human users – in AmGro-from-Home for example, individually modelling each member of the public would be far too fine-grained. In these cases, it is better to group all similar users together into one user (such as "U_The public" for AmGro), together with a declaration of how many of them there are. But, while this opposite extreme may also work in some cases, it may end up being too coarse grained - leading to a need to model different sorts of people, such as modelling those who have an established relationship with AmGro compared to those that do not as two sets of users "U_Registered Customer" and "U_Guest". In any case, it may be helpful to declare individual users to be "similar" when they could all be described by the same role or common set of roles (i.e. they can all be thought of as behaving in the same way or ways, and therefore all can be mapped to the same actor or set of actors).

Thus this "similar users" approach refines the notion to be "*a set of users of an IT system is defined by a representative user, together with a statement on their number*". Vitally, this refinement does not affect the notion of "actor" being a user in a role, although it requires a subtle change of language to the definition: "*a specific role of a collection of groups of users, and is referred to as an Actor*". It also does not alter the many-to-many relationship between user and actor.

This "representative user in multiple roles" approach has been adopted throughout this paper – it works well for modelling single systems, where the users and actors are identified from the unique perspective of their relationship(s) with the one system context (i.e. the scope of the model of the target IT system). But it must be adopted with care when modelling multiple systems.

[38] It is also valid to identify groups of users who all "have the same characteristics", whatever it is that they may be doing to (or for) this and any other IT System. This notion is popular in Enterprise Architecture, and is modelled as "User Groups".

Modelling across Multiple Systems

Modelling across multiple systems, particularly where there is an expectation that the same users will use more than one of the systems (as employees may in any enterprise) requires co-ordination between the individual teams designing each IT system. How this co-ordination is achieved depends on the nature of the systems' relationships:

- **Peer Systems**, where each system is broadly independent but needs to support a common set of groups of users (such as "Amgro-from-Home" and "AmGro Time-and-attendence" systems both interacting with a user "U_Call Centre Representative").

- **A hierarchy of systems**, in which a set of (sub)systems form part of a bigger whole that is itself being modelled as a system (such as modelling the whole "AmGro-from-Home" IT system as well as its part "AmGro-from-Home Order Management").

- **"Systems of systems"**, which, put simply, requires the modelling of user "A" interacting with system "B" (when designing system B) as well as user "B" interacting with system "A" (when designing system A) (for example, "A" might be the AmGro-from-Home order management system, and "B" the AmGro-from-Home warehouse management system).

The first two of these is discussed in detail below, the third ("Systems of Systems") is discussed in section "Appendix B: Modelling a System of Systems". In all three cases remember that a user, whether considered to be an individual or representative can be a real person or an external IT System.

Peer Systems: requiring a common set of definitions

Modelling an environment in which a user independently uses multiple target IT systems (such as AmGro-from-Home's order management system and the associated warehouse management system) requires each target IT system to model their users and roles in the same way – for example, AmGro-from-Home's Order Management and Warehouse Management projects will each want to model their system's relationship[39] with the same user "U_Customer" to help ensure AmGro's customer sees "one AmGro". Sometimes, the user will be in the same role[40] (such as "A_Home Shopper") but the user may also act in different ways with each system (such as "A_Complaining Shopper" for the order management project and "A_Delivery Arranger" for the warehouse management project).

In these cases, it may be advantageous for the programme overseeing the two projects to declare a single set of user and actor definitions (and maybe even a "standard set" of access mechanisms and exchanges), so that each project can be confident they will co-exist within the same wider environment. Since this co-operation applies at all levels, such as between many programmes within an enterprise, this "building block" approach is often systematically managed via an enterprise's Enterprise Architecture, defining, organising and publishing these common definitions via the notion of sets of ABBs (Architecture Building

[39] While the user may be the same, their relationship with each system may be different – leading to the user using system specific physical access mechanisms and interacting through different physical exchanges.
[40] While the actor may be the same, their relationship with each system may be different – leading to the actor using system specific logical access mechanisms and interacting through different logical exchanges.

Blocks), one type of ABB for each type of model element (users, actors, access mechanisms, exchanges and so on).

Hierarchy of Systems: basing the sub-system's users on the system's actors

Modelling users and actors across a hierarchy of systems (in which each sub-system is focused on part of the scope of the super-system) leads to the need for users' and actors' definitions to be increasingly specialised, the degree of specialisation driven by the system's context. This super/sub relationship can be multi-layered, so that a sub-system's scope is further divided, or the super-system is actually a part of something bigger.

In these circumstances, it will be helpful to refine any given level's "representative roles" into finer sub-roles at the lower level, with a decreasing diversity of users/actors represented as the scope of a model reduces level by level given the increasing degree of specialisation down the hierarchical structure. Then, individual system models can adopt a particular level and focus within this open ended hierarchy, in order to reason about appropriate "user sets" acting in various but specific ways, each way described as "an actor" from the perspective of the focus system.

For example, and using the prefix "U_" for user and "A_" for actor, consider the hierarchical family of "systems" illustrated in Figure 20:

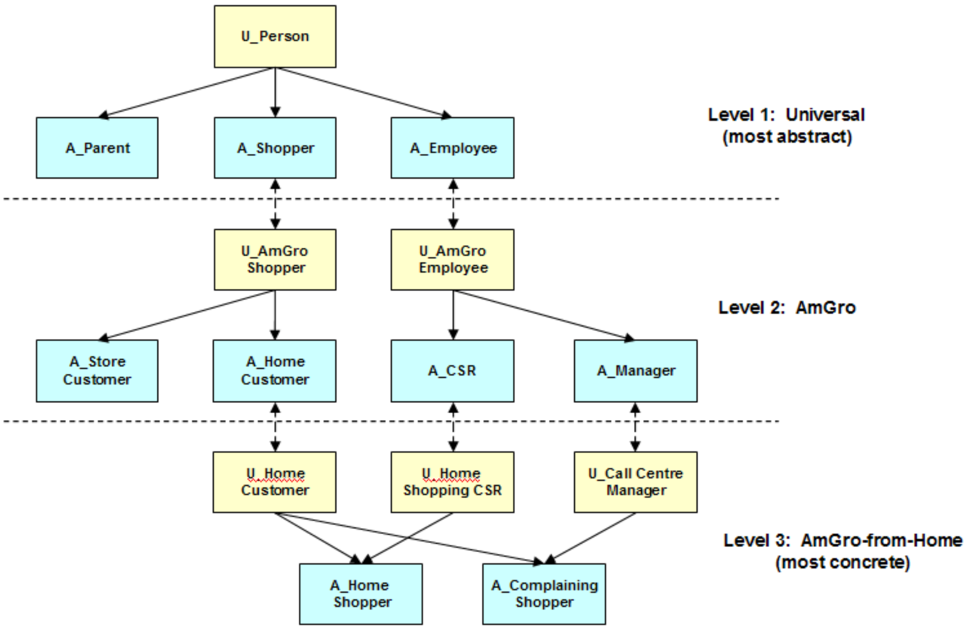


Figure 20: Hierarchy of Users and Actors

Figure 20 shows a (stylised) three level hierarchy:

- **Level 1 (Universal)**
  At this level, the system is "everything and everyone" for which every individual person (or IT System) may be modelled individually, such as **U_Person**. Given the breadth of the level, every individual will have a very broad range of roles, albeit defined in some generic fashion so that many individuals can have the same role. In Figure 20, one of these generic roles is **A_Employee**: each person may be an employee of one or more companies, and there will be many companies and

employees - but in all of them each person fills the same generic role, A_Employee.

It happens that one of the companies a U_Person works for is AmGro.

- **Level 2 AmGro**
  A more specific context is the enterprise called AmGro. As discussed earlier it's convenient for AmGro to model it's systems around sets of people rather than individuals – and one (large) set of individuals who are all more-or-less the same is its employees. Therefore, and specifically within the context of AmGro, a user (group) **U_AmGro Employee** has been defined.

  Critically, this is broadly equivalent (in functional terms) to the upper level's role "A_Employee" – hence the dotted double headed arrow. However, there are clearly differences (hence the word "broadly") between Level 1's A_Employee level 2's U_employee below: for example there may be millions of A_Employees[41], but far fewer (many 100s?) AmGro U_Employees; while other data associated with U_AmGro Employee may be more specialised than that of the corresponding broader context actor A_Employee (since it's related to far fewer people).

Thus:

- **Level 1: Universal**
  - **User:** An individual human "*U_Person*", who may adopt many roles
  - **Actor:** These roles can include "*A_Parent*", "*A_Shopper*" and "*A_Employee*".

- **Level 2: AmGro**
  - **Users**: the role "*A_Parent*" is not interesting. The other roles warrant further refinement:
    - Of those people who adopt the role of "A_Shopper", some can be modelled as people shopping at AmGro, "*U_AmGro Shopper*".
    - Of those people who adopt the role of "A_Employee", some can be modelled as AmGro's employees "*U_AmGro Employee*"

  - **Actor**: "U_AmGro Shopper" adopts the roles of "*A_Store Customer*" and "*A_Home Customer*", while "U_AmGro Employee" adopts the roles of "*A_CSR*" and "*A_Manager*"

- **Level 3: AmGro-from-Home**
  - **User**: the role "A_Store Customer" is not interesting, while the other roles warrant further refinement:
    - Those people who adopt the role of "A_Home Customer", are modelled as "*U_Customer*".
    - Of those people who adopt the roles of "A_CSR" and "A_Manager" some (i.e. those working on AmGro-from-Home) are modelled as users "*U_Home Shopping CSR*" and "*U_Call Centre Manager*"

[41] A different perspective would be to observe any U_Person will have very few, often only one A_Employee.

  - **Actor**: in varying combinations, these three users adopt the roles of "*A_Home Shopper*" and "*A_Complaining Shopper*"

Hence, it is often necessary to not only model a target IT system's "users" and "actors" in a manner that is sensitive to the scope of the target IT System, but also in a way that is aware of the wider context in which the target system sits – a project may only interested in those groups of users (human and IT systems) and their roles using the project's target IT system, but it must also model them in sympathy with the other system those users are interacting with.

In both cases ("Peer systems" ands "Hierarchy of Systems") there is no need to take account of systems interacting – that is, the situation in which one target system is a user of another, and vice versa. Handling actors and users when modelling "Systems of Systems" is discussed in section "Appendix B: Modelling a System of Systems".

Appendix B: Modelling a System of Systems

This paper focuses almost exclusively on system context modelling of a *single target IT Systems, in which all the constituent parts inside the target system are information technology*. All Humans have therefore been modelled as outside this single target IT system, whether as actors (logically) or users (physically).

However, the deployment of any given target IT system is rarely “stand alone” – for example, there are at least two ways of in which the “AmGro-from-Home Order Management” IT system is co-dependent on other “target systems”:

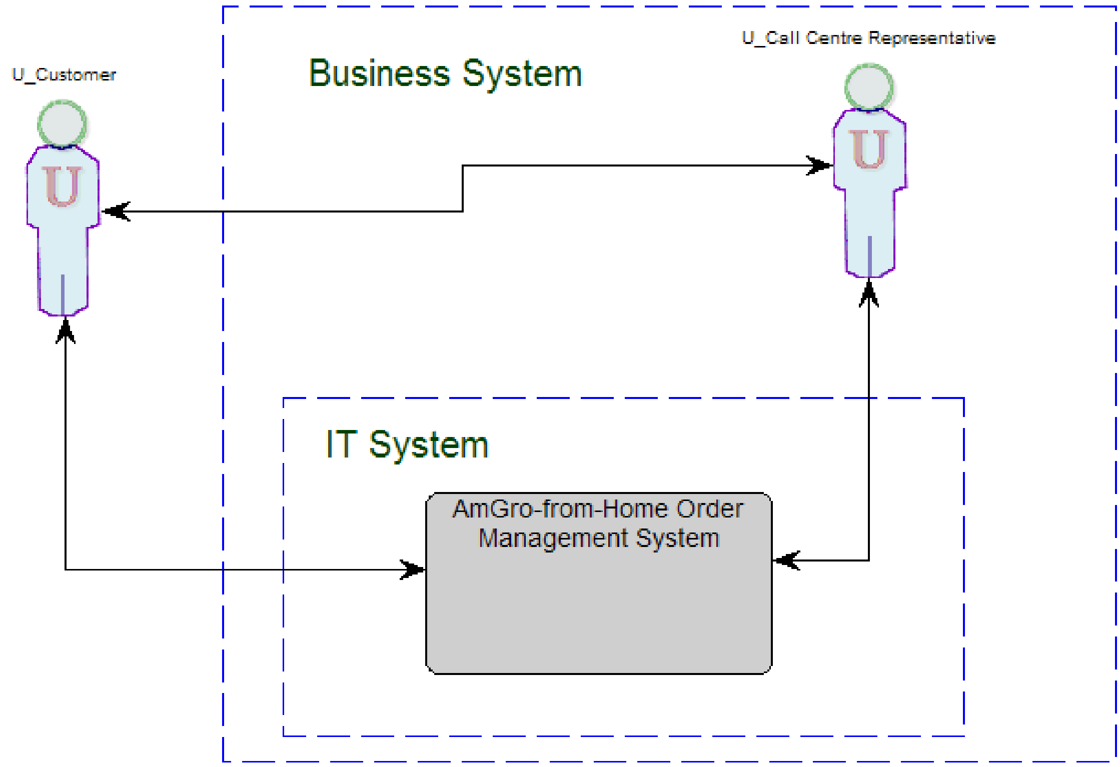


Figure 21: target IT System within a target business system

- Figure 21 shows it to be part of a bigger “AmGro-from-Home Order Management *Business* System”, which is being used by AmGro’s customers (i.e. they are *external* to the order management system) in two ways:
    - **electronically** (that is, the customers are users of the Order Management IT system)
    - **by telephone** to the Business System’s *internal* Call Centre Representatives (who are also users of the Order Management IT system)

  In this case, the Call Centre Representative may also be thought of as a “target system” and therefore in need of modelling in two ways – as an external user of the target IT system (U_Call Centre Representative), but also as an internal part of the business system.

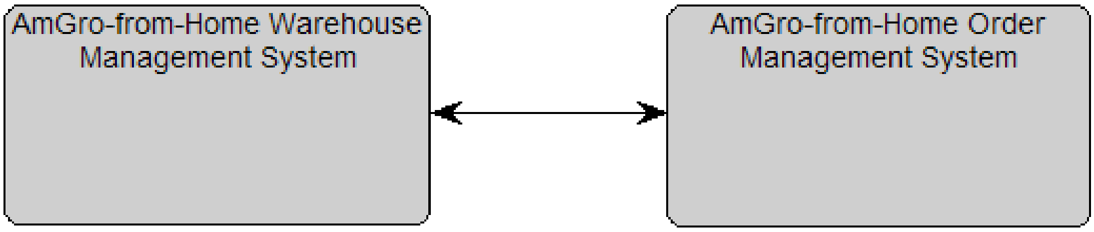


Figure 22: two interacting target IT systems

- Figure 22 shows it to be working in concert with another "target IT system" responsible for the warehouse management aspects of AmGro-from-Home. In this case, it is necessary to model the warehouse management system as a user of the order management system:

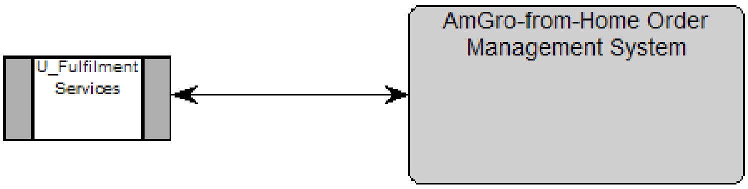


Figure 23: System A as a user of System B

as well as the order Management system as a user of the warehouse management system:

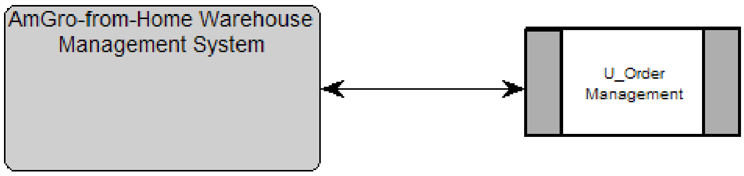


Figure 24: System B as a user of System A

"Appendix A: Users and Actors" addressed two relatively simple examples of this influence, in which the target system was in some way related to a wider world in which it's users and actors also needed to use other systems. However, these were cases in which the target system was modelled in isolation, with only a need to have a shared understanding of its external agents.

The following section discusses modelling these **"Systems of systems"** through an analysis of Figure 21.

Modelling the target IT System within a target Business System

Figure 21 shows a simplified but typical situation in which some of a target IT system's "external" users – whether they be human or external IT systems – will almost always be "internal" to the target business system *These internal users are therefore part of the overall target system being deployed for use by its ultimate end users*.

In order to model this distinction, humans considered part of the Business System are referred to as "workers". This helps, for example, with modelling the skills and expectations of those using the target IT System, since workers can be expected to undergo greater amounts of training and education, tolerate different levels of support and/or performance, or require different Human-Machine interfaces (such as a scripted user interface).

Further, it may be that the decision to use "internal users" rather than implementing that aspect of the target system in IT is not something which can be readily determined from the outset, in which case the decision to implement part of the logical design of the target system as a (human) worker rather than an (IT) physical component is taken sometime later in the design process.

In order to model this, it is necessary to "look inside" the solution design, as shown in

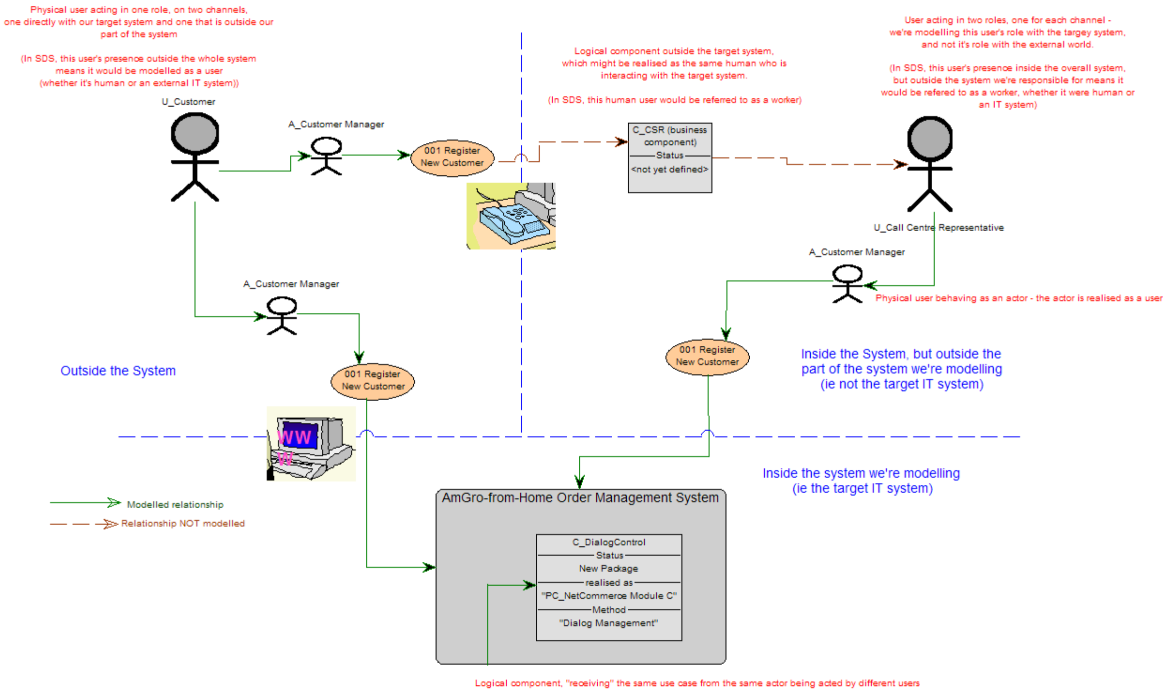


Figure 25: Sketching the insides of a Business System

The example in Figure 25 is taken from AmGro-from-Home, and shows two different users of the target IT System acting in the same role (A_Customer Manager) initiating the same Use Case with the target system (001 Register New Customer). In turn, the target system is interacting with this (one) actor via a logical component C_DialogControl.

The figure also shows user U_Customer acting in the same A_Customer Manager role, doing the same thing, but via a different access mechanism – the telephone (not shown). In this case they are interacting with the Business System – something that's not been modelled in the AmGro-from-Home Order Management IT system.

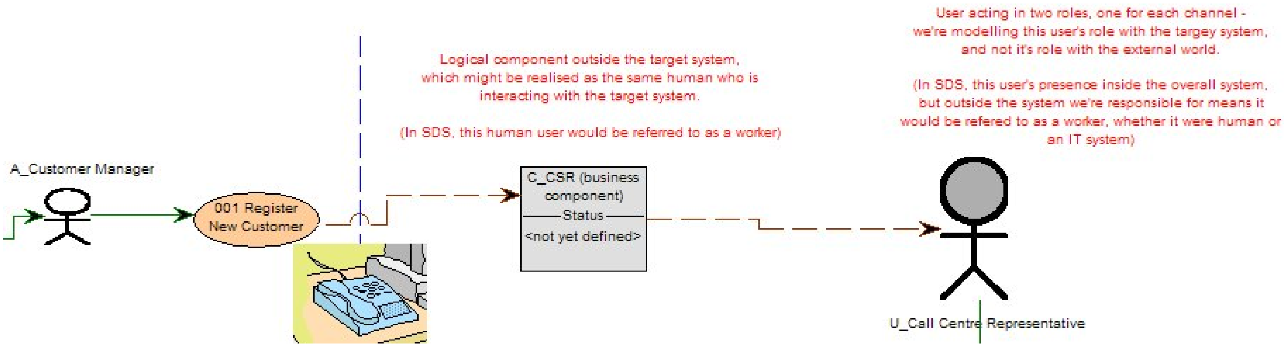


Figure 26: Sketching the insides of a Business System (detail)

However, as shown in the close up Figure 26 it would be perfectly possible to model this in exactly the same way as modelling the customer's interaction with the target IT system, via a logical component called C_CSR.

As the status of the logical component C_CSR says, the implementation choice for this logical component has not yet been made, but it could be implemented either as a human, or in IT:

- <u>If it were decided to implement this logical component as a person</u> *(represented by the dotted arrow),* then the logical component's realisation would be as user "U_Call Centre Representative", which would, in fact, then be stereotyped as a user of type *worker*.[42] Consequentially, the user U-Call Centre Representative would be associated with two distinctly different logical descriptions of its role – one as an actor (A_Customer Manager), and one as a logical component (C_CSR)[43].

- <u>If it were decided to implement this logical component in IT</u> *(not shown),* then the logical component's realisation would be as a physical component "PC_Voice Response Unit" (or similar!), which would (in all probability) then remain within the target IT System.

Finally, it may be that the logical component C_CSR's realisation will be in an existing IT system (maybe a pre-existing Automated Call Director system). In this case, it would be necessary to model this as user U_ACD for the target IT system.

[42] SDS models "worker" as a stereotype of "physical component". Either would be acceptable, since the physical U_Call Centre Representative is behaving as a logical actor and a logical component.
[43] This capability to model workers has not been implemented in SA4TeamSD.

Appendix C: Borders and Boundaries

Care may be required to separate the notion of "boundary" from "border". The reader is referred to [Ref 1] for a detailed distinction, but in simple terms:

- A boundary has two sides and no thickness. It is between two regions and has no properties other than to document the difference in some characteristic of the elements in each region either side of it.
- A border is a model element in its own right, being the thing that connects locations. It therefore has any number of properties of its own, many of which are the same type as those of location.

Examples can be found from various parts of the real world:

- **Sports:** Many sports pitches have a line around the perimeter – this is the pitch's border. But the location of the boundary can vary depending on the sport being played: for some, the boundary is the inside edge of the white line – the ball or player is "out" the moment it/they touch the line; whereas for other sports the boundary is the outside edge – the ball or player is only "out" when all of it/them have fully crossed the line. In all cases, there are two regions: "in" and "out".

- **Politics:** A nation's border with a neighbouring country often has customs posts on either side, with some form of "no man's land" in between. In modelling terms, the customs posts are the border's boundaries, separating three regions: Nation A, No-man's land, and Nation B.

Ref 1 defines a set of things that are "all the same side of a boundary" to be in a **zone**, with boundaries identifying where one zone ends and another begins. We commonly model locations and borders to be in zones, in order to highlight a change in some non functional characteristic such as systems management capability, safety level, or, as is the case in Figure 27, different security levels:

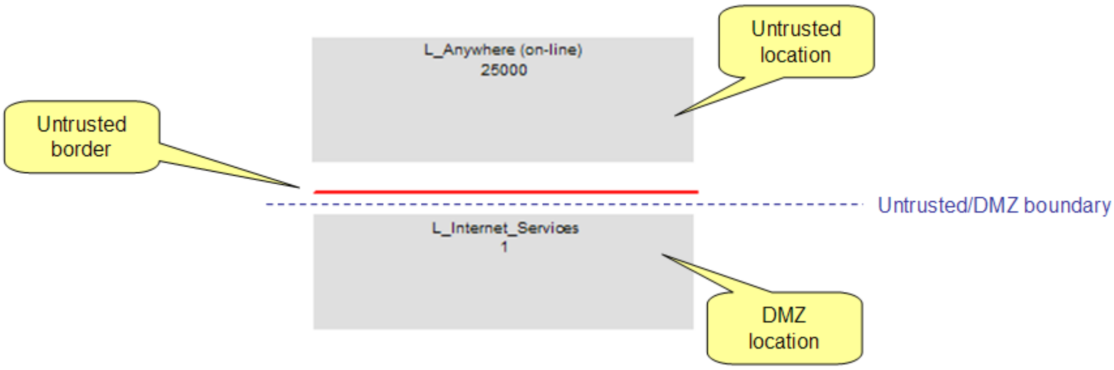


Figure 27: Location, border and boundary

Here, the location L_Anywhere (on-line) and the border have been classified as "untrusted" (they make up the "untrusted zone") while L_Internet Services is a "demilitarised zone". In this case, therefore, the boundary lies between the untrusted border and the DMZ Location.

References

1. *"System Description Standard R3: Semantic Specification"*; Philippe Spaas, https://w3.tap.ibm.com/w3ki/display/SDSR3/Home

2. *"An Introduction to the IBM Views and Viewpoints Framework for IT Systems"*; Denise Cook, Peter Cripps, Philippe Spaas http://www.ibm.com/developerworks/rational/library/08/0108_cooks-cripps-spaas/

3. *"Team Solution Design White Paper: 'Component Modelling'"* To be written

4. *"Team Solution Design White Paper: 'Operational Modelling'"* To be written

