Resumen Información sobre Caritas

- El objetivo principal de Cáritas, según sus estatutos es “animar y coordinar la obra social y caritativa de la Iglesia, insertada en la pastoral orgánica a través de formas adaptadas al tiempo y las circunstancias, para lograr el desarrollo integral de todo el hombre y de todos los hombres, con especial preferencia por las personas y comunidades más marginadas“.
- A mediados de la década del ´50 comienza la labor de Cáritas en Argentina. Organización con 70 años de antigüedad.
- En la actualidad, Cáritas Argentina se encuentra trabajando activamente en las 67 Diócesis de la Iglesia Argentina. Gracias al compromiso solidario de toda la sociedad y al esfuerzo cotidiano de más de 32.000 voluntarios, Cáritas canaliza su acción a través de más de 3.500 parroquias, capillas y centros misionales.
- Nos centramos en Cáritas Nacional.
- CÁRITAS PARROQUIALES
    - Alrededor de 3.500 parroquias, capillas y centros misionales, animan y coordinan en todo el país la tarea de Cáritas en su ámbito local. Su acción llega de manera directa a las familias más pobres de cada comunidad.
- CÁRITAS DIOCESANAS
    - Animan, articulan y coordinan la tarea de las Cáritas parroquiales y llevan adelante programas y proyectos de asistencia y promoción a nivel diocesano y regional.
- CÁRITAS NACIONAL
    - Anima y apoya la acción de las Cáritas diocesanas articulando y coordinando recursos, programas y esfuerzos mediante el acompañamiento, la asistencia técnica, la capacitación y el monitoreo de sus equipos de trabajo.
- **Las 3 formas de asistencia de Cáritas:**
    - **Asistencia:** Forma de animar la caridad que consiste en dar respuestas a las necesidades mínimas o situaciones de emergencia de las comunidades más pobres.
    - **Promoción humana:** Busca modificar, mejorar y suscitar cambios que mejoren las condiciones de vida de los más pobres. Requiere incorporar a las personas en la búsqueda de soluciones a sus problemas junto con otros.
    - **Caridad transformadora:** La Iglesia desde la búsqueda de un mundo de hermanos, apunta a transformar las estructuras injustas de pecado y producir cambios en toda la sociedad para acercarnos al Proyecto de Dios.

- **Tipos de Donaciones que recibe Cáritas:**
    - **Dinero:** Brindan montos de referencia aunque permiten aportar el importe que uno vea necesario o pueda, cuentan con mecanismos de pago ya incorporados (pago con crédito, débito y mercado pago) y además aceptan donaciones del exterior desde HELPARGENTINA.
    - **Especie**: Requiere acercarse a una sede de Cáritas Parroquial para donar especies
        - Reciben **ropa** necesariamente limpia, completa y que no le falten botones o cierres en bolsas o cajas clasificadas para hombre, mujer o niño.
        - Reciben **alimentos** controlando la fecha de vencimiento y que los envases estén sellados herméticamente de fábrica.
        - Reciben **medicamentos** en muy pocas Cáritas parroquiales.
    - **Legados Solidarios:** Consta en destinar una parte de tu herencia a una organización social, en este caso Cáritas, con el objetivo de mejorar la vida de las personas en situación de mayor vulnerabilidad y brindarles más oportunidades. Para esto se debe incluir a Cáritas en el testamento de la persona. Se pueden legar bienes inmuebles (departamentos, casas, terrenos, etc.) y bienes muebles (dinero, acciones, joyas, vehículos, mobiliario, etc.). Podés legar hasta un tercio de tus bienes y este legado solo se hace efectivo luego del fallecimiento de una persona.
- **¿Quiénes pueden donar?**
    - **Personas**: Toda persona que quiera sumarse a nuestro desafío puede colaborar a través de donaciones en dinero, especie o eventualmente, un legado.
    - **Organizaciones u Empresas:** En Cáritas promovemos alianzas con organizaciones, empresas e instituciones para que, animadas por el sentido de la solidaridad, se involucren, colaboren y se comprometan en la construcción de un mundo más justo.

Información Adicional sobre Ingeniería

- None of these requirements specify how the system is to be implemented. There is no mention of what database-management system to use, whether a client-server architecture will be employed, how much memory the computer is to have, or what

programming language must be used to develop the system. These implementation-specific descriptions are not considered to be requirements (unless they are mandated by the customer).The goal of the requirements phase is to understand the customer's problems and needs. Thus, requirements focus on the customer and the problem, not on the solution or the implementation. We often say that requirements designate what behavior the customer wants, without saying how that behavior will be realized. Any discussion of a solution is premature until the problem is clearly defined.

Traducción: Ninguno de estos requisitos especifica cómo se implementará el sistema. No se menciona qué sistema de gestión de bases de datos se utilizará, si se empleará una arquitectura cliente-servidor, cuánta memoria deberá tener la computadora, ni qué lenguaje de programación se debe utilizar para desarrollar el sistema. Estas descripciones específicas de implementación no se consideran requisitos (a menos que sean exigidos por el cliente). El objetivo de la fase de requisitos es comprender los problemas y necesidades del cliente. Por lo tanto, los requisitos se centran en el cliente y el problema, no en la solución o la implementación. A menudo decimos que los requisitos indican qué comportamiento quiere el cliente, sin especificar cómo se realizará ese comportamiento. Cualquier discusión sobre una solución es prematura hasta que el problema esté claramente definido.

- So who are the stakeholders? It turns out that there are many people who have something to contribute to the requirements of a new system:
    - **Clients**, who are the ones paying for the software to be developed: By paying for the development, the clients are, in some sense, the ultimate stakeholders, and have the final say about what the product does (Robertson and Robertson 1999).
    - **Customers**, who buy the software after it is developed: Sometimes the customer and the user are the same; other times, the customer is a business manager who is interested in improving the productivity of her employees.We have to understand the customers' needs well enough to build a product that they will buy and find useful.
    - **Users**, who are familiar with the current system and will use the future system: These are the experts on how the current system works, which features are the most useful, and which

aspects of the system need improving. We may want to consult also with special-interest groups of users, such as users with disabilities, people who are unfamiliar with or uncomfortable using computers, expert users, and so on, to understand their particular needs.

- **Domain experts**, who are familiar with the problem that the software must auto-mate: For example, we would consult a financial expert if we were building a financial package, or a meteorologist if our software were to model the weather. These people can contribute to the requirements, or will know about the kinds of environments to which the product will be exposed.
- **Market researchers,** who have conducted surveys to determine future trends and potential customers' needs: They may assume the role of the customer if our software is being developed for the mass market and no particular customer has been identified yet.
- **Lawyers or auditors, who are familiar with government, safety, or legal requirements**: For example, we might consult a tax expert to ensure that a payroll package adheres to the tax law.We may also consult with experts on standards that are relevant to the product's functions.
- **Software engineers or other technology experts:** These experts ensure that the product is technically and economically feasible. They can educate the customer about innovative hardware and software technologies, and can recommend new functionality that takes advantage of these technologies. They can also estimate the cost and development time of the product

Traducción: Entonces, ¿quiénes son los interesados? Resulta que hay muchas personas que tienen algo que contribuir a los requisitos de un nuevo sistema:

- *Clientes, quienes son los que pagan por el desarrollo del software: Al pagar por el desarrollo, los clientes son, en cierto sentido, los principales interesados finales, y tienen la última palabra sobre lo que hace el producto (Robertson y Robertson 1999).*
- Consumidores*, que compran el software después de que se desarrolle: A veces el cliente y el usuario son los mismos; otras veces, el cliente es un gerente de negocios interesado en mejorar la productividad de sus empleados. Debemos entender las necesidades de los clientes lo suficientemente*

bien como para construir un producto que ellos comprarán y encontrarán útil.

- *Usuarios, quienes están familiarizados con el sistema actual y usarán el sistema futuro: Ellos son los expertos en cómo funciona el sistema actual, cuáles son las características más útiles y qué aspectos del sistema necesitan mejorar. También puede ser útil consultar con grupos de interés especial de usuarios, como usuarios con discapacidades, personas que no están familiarizadas o se sienten incómodas usando computadoras, usuarios expertos, y así sucesivamente, para entender sus necesidades particulares.*
- *Expertos en el dominio, quienes están familiarizados con el problema que el software debe automatizar: Por ejemplo, consultaremos a un experto financiero si estuviéramos construyendo un paquete financiero, o a un meteorólogo si nuestro software fuera a modelar el clima. Estas personas pueden contribuir a los requisitos, o conocerán los tipos de entornos a los que el producto estará expuesto.*
- *Investigadores de mercado, quienes han realizado encuestas para determinar las tendencias futuras y las necesidades potenciales de los clientes: Pueden asumir el rol del cliente si nuestro software se está desarrollando para el mercado masivo y aún no se ha identificado a un cliente en particular.*
- *Abogados o auditores, quienes están familiarizados con los requisitos gubernamentales, de seguridad o legales: Por ejemplo, podríamos consultar a un experto en impuestos para asegurarnos de que un paquete de nómina cumpla con la ley fiscal. También podríamos consultar con expertos en estándares que sean relevantes para las funciones del producto.*
- *Ingenieros de software u otros expertos en tecnología: Estos expertos garantizan que el producto sea técnicamente y económicamente viable. Pueden educar al cliente sobre tecnologías innovadoras de hardware y software, y pueden recomendar nuevas funcionalidades que aprovechen estas tecnologías. También pueden estimar el costo y el tiempo de desarrollo del producto.*

- A **functional requirement** describes required behavior in terms of required activities, such as reactions to inputs, and the state of

each entity before and after an activity occurs. For instance, for a payroll system, the functional requirements state how often paychecks are issued, what input is necessary for a paycheck to be printed, under what conditions the amount of pay can be changed, and what causes the removal of an employee from the payroll list. A functional requirement describes required behavior in terms of required activities, such as reactions to inputs, and the state of each entity before and after an activity occurs. For instance, for a payroll system, the functional requirements state how often paychecks are issued, what input is necessary for a paycheck to be printed, under what conditions the amount of pay can be changed, and what causes the removal of an employee from the payroll list.

Traducción: Un requisito funcional describe el comportamiento requerido en términos de actividades necesarias, como reacciones a entradas y el estado de cada entidad antes y después de que ocurra una actividad. Por ejemplo, para un sistema de nómina, los requisitos funcionales establecen con qué frecuencia se emiten los cheques de pago, qué entrada es necesaria para que se imprima un cheque de pago, bajo qué condiciones se puede cambiar la cantidad de pago y qué causa la eliminación de un empleado de la lista de nómina.

- A **quality requirement, or nonfunctional requirement,** describes some quality characteristics that the software solution must possess, such as fast response time, ease of use, high reliability, or low maintenance costs. A design constraint is a design decision, such as choice of platform or interface components, that has already been made and that restricts the set of solutions to our problem.A process constraint is a restriction on the techniques or resources that can be used to build the system. For example, customers may insist that we use agile methods, so that they can use early versions of the system while we continue to add features. Thus, quality requirements, design constraints, and process constraints further restrict our solution space by differentiating acceptable, well-liked solutions from unused products. Table 4.2 gives examples of each kind of requirement. Quality requirements sometimes sound like "motherhood" characteristics that all products ought to possess. After all, who is going to ask for a slow, unfriendly, unreliable, unmaintainable software system? It is better to think of quality requirements as design criteria that can be optimized and can be used to choose among alternative implementations of functional requirements. Given this approach, the question to be answered by the requirements is: To what extent

must a product satisfy these quality requirements to be acceptable?

Traducción: Un requisito de calidad, o requisito no funcional, describe algunas características de calidad que la solución de software debe poseer, como tiempo de respuesta rápido, facilidad de uso, alta confiabilidad o bajos costos de mantenimiento. Una restricción de diseño es una decisión de diseño, como la elección de la plataforma o los componentes de la interfaz, que ya se ha tomado y que restringe el conjunto de soluciones a nuestro problema. Una restricción de proceso es una restricción sobre las técnicas o recursos que se pueden utilizar para construir el sistema. Por ejemplo, los clientes pueden insistir en que utilicemos métodos ágiles, para que puedan usar versiones tempranas del sistema mientras continuamos agregando características. Por lo tanto, los requisitos de calidad, las restricciones de diseño y las restricciones de proceso restringen aún más nuestro espacio de soluciones al diferenciar soluciones aceptables y bien recibidas de productos no utilizados. La tabla 4.2 da ejemplos de cada tipo de requisito. Los requisitos de calidad a veces suenan como características básicas que todos los productos deben poseer. Después de todo, ¿quién va a pedir un sistema de software lento, poco amigable, poco confiable e inmantenible? Es mejor pensar en los requisitos de calidad como criterios de diseño que se pueden optimizar y se pueden utilizar para elegir entre implementaciones alternativas de requisitos funcionales. Dado este enfoque, la pregunta que debe responderse mediante los requisitos es: ¿Hasta qué punto debe satisfacer un producto estos requisitos de calidad para ser aceptable?

TABLE 4.1 How Users and Developers View Each Other (Scharer 1990)

How Developers See Users	How Users See Developers
Users don't know what they want.	Developers don't understand operational needs.
Users can't articulate what they want.	Developers can't translate clearly stated needs into a successful system.
Users are unable to provide a usable statement of needs.	Developers set unrealistic standards for requirements definition.
Users have too many needs that are politically motivated.	Developers place too much emphasis on technicalities.
Users want everything right now.	Developers are always late.
Users can't remain on schedule.	Developers can't respond quickly to legitimately changing needs.
Users can't prioritize needs.	Developers are always over budget.
Users are unwilling to compromise.	Developers say "no" all the time.
Users refuse to take responsibility for the system.	Developers try to tell us how to do our jobs.
Users are not committed to development projects.	Developers ask users for time and effort, even to the detriment of the users' important primary duties.

TABLE 4.2 Questions to Tease Out Different Types of Requirements

Functional Requirements

Functionality

- What will the system do?
- When will the system do it?
- Are there several modes of operation?
- What kinds of computations or data transformations must be performed?
- What are the appropriate reactions to possible stimuli?

Data

- For both input and output, what should be the format of the data?
- Must any data be retained for any period of time?

Design Constraints

Physical Environment

- Where is the equipment to be located?
- Is there one location or several?
- Are there any environmental restrictions, such as temperature, humidity, or magnetic interference?
- Are there any constraints on the size of the system?
- Are there any constraints on power, heating, or air conditioning?
- Are there constraints on the programming language because of existing software components?

Interfaces

- Is input coming from one or more other systems?
- Is output going to one or more other systems?
- Is there a prescribed way in which input/output data must be formatted?
- Is there a prescribed medium that the data must use?

Users

- Who will use the system?
- Will there be several types of users?
- What is the skill level of each user?

Process Constraints

Resources

- What materials, personnel, or other resources are needed to build the system?
- What skills must the developers have?

Documentation

- How much documentation is required?
- Should it be online, in book format, or both?
- To what audience should each type of documentation be addressed?

Standards

Quality Requirements

Performance

- Are there constraints on execution speed, response time, or throughput?
- What efficiency measures will apply to resource usage and response time?
- How much data will flow through the system?
- How often will data be received or sent?

Usability and Human Factors

- What kind of training will be required for each type of user?
- How easy should it be for a user to understand and use the system?
- How difficult should it be for a user to misuse the system?

Security

- Must access to the system or information be controlled?
- Should each user's data be isolated from the data of other users?
- Should user programs be isolated from other programs and from the operating system?
- Should precautions be taken against theft or vandalism?

Reliability and Availability

- Must the system detect and isolate faults?
- What is the prescribed mean time between failures?
- Is there a maximum time allowed for restarting the system after a failure?
- How often will the system be backed up?
- Must backup copies be stored at a different location?
- Should precautions be taken against fire or water damage?

Maintainability

- Will maintenance merely correct errors, or will it also include improving the system?
- When and in what ways might the system be changed in the future?
- How easy should it be to add features to the system?
- How easy should it be to port the system from one platform (computer, operating system) to another?

Precision and Accuracy

- How accurate must data calculations be?
- To what degree of precision must calculations be made?

Time to Delivery / Cost

- Is there a prescribed timetable for development?
- Is there a limit on the amount of money to be spent on development or on hardware or software?