Escola Tècnica Superior d'Enginyeria ETSE-UV

GRADO EN INGENIERÍA MULTIMEDIA

VNIVERSITAT DE VALÈNCIA

TRABAJO DE FIN DE GRADO

CONVERSOR DE MALLAS 3D DE FIBRAS DE PURKINJE PARA EL MODELADO DEL CORAZÓN EN CARP

AUTOR:
VÍCTOR GUILLERMO ANDRÉS ESCUDERO

TUTOR:
FERNANDO BARBER MIRALLES

NOVIEMBRE DE 2018

GRADO EN INGENIERÍA MULTIMEDIA

TRABAJO DE FIN DE GRADO

CONVERSOR DE MALLAS 3D DE FIBRAS DE PURKINJE PARA EL MODELADO DEL CORAZÓN EN CARP

AUTOR:
VÍCTOR GUILLERMO ANDRÉS ESCUDERO

TUTOR:
FERNANDO BARBER MIRALLES

TRIBUNAL:

PRESIDENT/A:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Resumen

A lo largo de este proyecto se va a realizar una aplicación capaz de convertir ficheros de VTK, una de las librerías gráficas más importantes actualmente, a ficheros reconocibles por CARP un simulador del corazón usado, entre otros, por la Universidad de Valencia para realizar estudios cardiacos.

Este programa analiza cualquier fichero codificado en el formato propio de VTK, y es capaz de extraer toda su información geométrica (puntos, elementos, atributos, etc.) y generar los ficheros correspondientes con el formato y sintaxis de CARP. Por una parte los ficheros que describen la malla poligonal que define la geometría del corazón, que son el fichero de nodos donde se indexan y encuentran las coordenadas de los puntos, y el fichero de elementos que especifica las caras y/o polígonos del modelo del corazón; y por otra parte los ficheros de fibras de Purkinje donde se encuentran las vías especializadas de conducción eléctrica del corazón.

También se ha tenido muy presente a lo largo de la realización del programa que tanto VTK como CARP son software en desarrollo activo por lo que sus funcionalidades pueden cambiar con el paso del tiempo. Por esto se ha tenido especial cuidado en hacer un programa flexible, fácil de mantener y modificar, para que pueda ser sencillo añadir nuevas funcionalidades, por ejemplo soporte para nuevos formatos de archivos, o adaptar el programa a los posibles cambios de los otros programas.

Índice

1 Introducción

A lo largo de estos últimos años es indudable que la tecnología ha adquirido un protagonismo inmenso en nuestra sociedad. Cada vez hay nuevos dispositivos inteligentes capaces de hacer nuestro día a día más cómodo y surgen nuevas tecnologías que dan paso a descubrimientos que no se creían posible.

En el campo de la medicina uno de los principales avances que ha permitido el uso de la tecnología de la información es el desarrollo de simuladores, capaces de replicar el funcionamiento de varios órganos del cuerpo humano, con una fidelidad suficiente para realizar estudios preliminares en los simuladores en lugar de los propios órganos, ahorrando tiempo, riesgos y dinero.

Sin embargo, estos simuladores aunque muy prometedores aún están en sus primeras iteraciones y la llegada de nuevos simuladores con nuevas funcionalidades implica a su vez la necesidad de nuevas herramientas auxiliares que faciliten su uso y la compatibilidad con programas existentes.

Por todo esto, este proyecto trata de investigar y desarrollar una aplicación que genere varios de los archivos de entrada necesarios para realizar una simulación cardiaca en CARP a partir de un fichero de VTK, una de las principales librerías gráficas usadas.

2 Motivación y Objetivos

2.1 Motivación

Las enfermedades cardiacas son, de acuerdo a la Organización Mundial de la Salud, las principales causas de mortalidad en el mundo. Este dato evidencia la importancia de estas enfermedades sobre nuestra salud.

Aunque el corazón sea un órgano muy estudiado por la medicina, este presenta algunas particularidades que dificultan su estudio. Por ejemplo, realizar experimentos sobre corazones vivos es muy complicado, tanto por que cualquier fallo puede tener consecuencias mortales para el sujeto, como por las técnicas invasivas que se tienen que emplear para acceder hasta este debido a su ubicación.

Y es aquí donde la importancia de los simuladores cardiacos se hace patente. Si se es capaz de crear un programa que simule de forma realista el funcionamiento de un corazón, se pueden realizar cualquier número de experimentos sin preocuparse por los daños causados al corazón. Esto abre la puerta al desarrollo de nuevas técnicas y conocimientos que pueden ayudarnos a combatir las enfermedades cardiacas.

La idea de realizar este proyecto surge de esta idea pero específicamente se debe a que en la Universidad de Valencia hay un grupo de investigación precisamente de simulación cardiaca. En el momento de empezar este TFG este grupo de investigación estaba empezando a migrar de un simulador cardiaco antiguo (Elvira) a CARP, por lo que muchos de los ficheros que empleaban en las simulaciones no podían ser reusados. Fue por esto que mi tutor, Fernando Miralles, me aconsejó realizar un conversor de ficheros CARP, principalmente de ficheros de fibras de Purkinje, aquellas fibras por las que la electricidad del corazón circula.

2.2 Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación capaz de crear los ficheros que usa el simulador CARP como datos de entrada para sus simulaciones a partir de ficheros guardados en formato .vtk usados por la librería gráfica homónima. Para realizar esto se han planteado los siguientes objetivos:

- Estudiar el funcionamiento del corazón, los avances en el campo de la simulación cardiaca y específicamente el simulador cardiaco CARP. De esta forma se adquirirán los conocimientos necesarios para poder emprender el resto de tareas desde una posición informada y saber cómo afrontar los posibles problemas que surjan.

- Crear un programa que cumpla con los principios básicos de la programación orientada a objetos (alta cohesión, bajo acoplamiento, etc.). Así conseguimos que el código de la aplicación sea de calidad y fácil de mantener por otros desarrolladores.

- Que la aplicación sea capaz no solo de convertir ficheros de VTK a CARP mediante una conversión 1:1, sino capaz también de adaptarse a los requisitos del simulador, añadiendo los datos necesarios y modificando cuando sea imperativo la malla en si para que cumpla con las especificaciones de los ficheros de entrada.

3 Estado el Arte

Para poder realizar con éxito el proyecto primero es necesario obtener unos conocimientos sobre el tema del que se va a realizar el proyecto, se investigaran los avances científicos que han ido ocurriendo a lo largo de los años, como se han superado las posibles limitaciones que han ido surgiendo, así como las técnicas y métodos que se han usado para poder aprender de ellas y poder afrontar el proyecto desde una posición informada, correcta, y actual.

3.1 El Corazón

El corazón es uno de los órganos más importantes del cuerpo humano, situado en el tórax, entre los pulmones, permite la circulación constante de la sangre por los vasos sanguíneos situados a lo largo del cuerpo.

3.1.1 Sistema Circulatorio

Es a su vez el órgano principal del sistema circulatorio, formado por el corazón, los vasos sanguíneos, la sangre y el sistema linfático. Se encarga de pasar nutrientes a las células del cuerpo, como el oxígeno, y recoger los desechos metabólicos [1].

La sangre que entra al corazón carbónica y con unos niveles pobres de oxígeno, es enviada por la arteria pulmonar hacía los pulmones donde se libera parte del CO_2 y se oxigena, esta sangre rica en nutrientes vuelve ser llevada al corazón donde se envía al resto del cuerpo. Las células usan entonces este oxígeno en la sangre para obtener energía y produciendo CO_2 que es depositado en la sangre para volver a ser transportada al corazón [2].

Esto se conoce como circulación sanguínea doble, ya que la sangre pasa dos veces por el corazón en cada vuelta. El circuito pulmonar es aquel que lleva la sangre desoxigenada hasta los vasos sanguíneos de los pulmones donde vuelve ya oxigenada al corazón. Y el circuito general, que lleva la sangre recientemente oxigenada al resto del cuerpo y vuelve al corazón desoxigenada. Y puesto que la sangre oxigenada nunca se mezcla con la desoxigenada, la circulación sanguínea también es completa.

3.1.2 Anatomía del corazón

El corazón siendo un órgano muy complejo y compuesto por diversas partes y estructuras. Conocerlas y entender cómo interactúan entre ellas es muy importante para entender su funcionamiento, y poder realizar simulaciones que se acerquen cada vez más a la realidad.

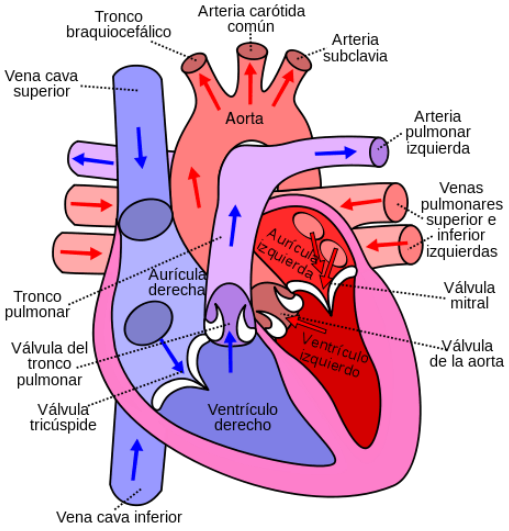

Ilustración 1. Partes del corazón

Estructura interna del corazón

Como se ha visto, la circulación sanguínea en el ser humano es un mecanismo muy intricado, esto es posible gracias a que el corazón presenta una estructura interna compleja formada por 3 elementos principales: las cámaras, tabiques y válvulas.

-Cámaras del corazón

Existen 4 cámaras en el corazón humano. Dos aurículas, derecha e izquierda, situadas en la parte superior y dos ventrículos en la inferior. Las aurículas actúan como recipientes de entrada de la sangre al corazón y se contraen para empujar la sangre a las cámaras inferiores, los ventrículos derecho e izquierdo, encargados de bombear a las otras partes del cuerpo.

El corazón se puede dividir también en dos partes que son el corazón derecho e izquierdo, compuesto por su aurícula y ventrículo respectivos. El corazón derecho trabaja con sangre desoxigenada y es el actor principal del circuito pulmonar. La aurícula derecha recibe la sangre de las venas cavas después de ser usada por el cuerpo y la envía al ventrículo derecho que a su vez la bombea por el tronco pulmonar hacía los capilares de los pulmones donde es oxigenada. El corazón izquierdo, parte del circuito general, trabaja por lo tanto con sangre oxigenada recibida por las venas pulmonares y tras pasarla a su ventrículo es enviada por la aorta al resto del cuerpo [3].

-Tabiques del corazón

Los tabiques son las paredes que dividen el corazón en las 4 cámaras, como se verá más adelante están compuestos por 2 capas distintas, miocardio y endocardio. Existen 3 tabiques:

- Tabique interauricular: separa la aurícula derecha de la izquierda.
- Tabique interventricular: separa el ventrículo derecho del izquierdo.

- Tabique atrioventricular: separa las aurículas de los ventrículos.

El tabique atrioventricular se diferencia por la existencia en él de 4 válvulas que permiten el flujo unidireccional de la sangre. Estas válvulas debilitan estructuralmente el tabique por lo que es reforzado por un tejido llamado esqueleto cardiaco, que incluye 4 anillos que rodean las 4 válvulas del corazón, también actúa como frontera para el sistema de conducción eléctrica del corazón [4].

-Válvulas cardiacas

Las válvulas cardiacas, como se ha visto, permiten el flujo en una sola dirección de la sangre por el corazón.

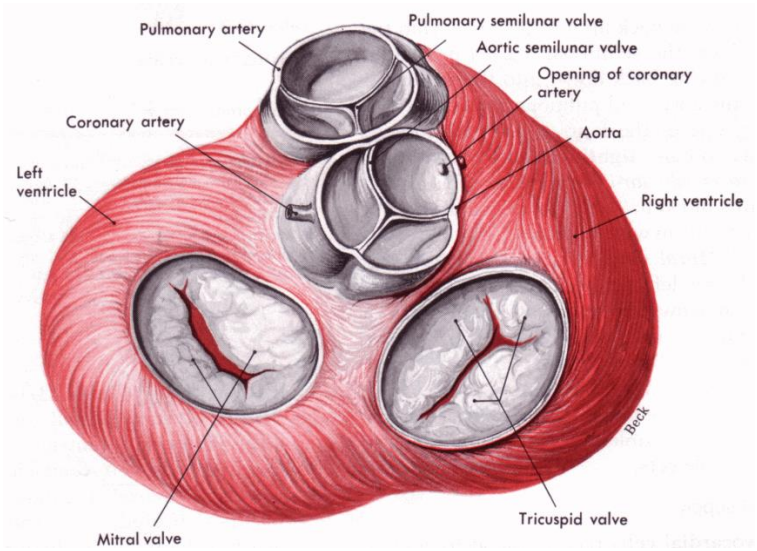

Ilustración 2. Válvulas cardiacas vistas desde arriba. Aurículas no mostradas.

La válvula del tronco pulmonar (o válvula pulmonar) evita que entre sangre en el ventrículo derecho por el tronco pulmonar, mientras que la válvula de la aorta (o válvula aórtica) evita que entre sangre en el ventrículo izquierdo por la aorta.

Estas válvulas impiden el reflujo de la sangre gracias la forma de sus solapas cóncavas (ilustración 2).

Las dos válvulas restantes son la válvula tricúspide y válvula mitral (o válvula bicúspide) que impiden la reentrada de la sangre a las aurículas derecha e izquierda, respectivamente, desde sus ventrículos. Son las válvulas atrioventriculares.

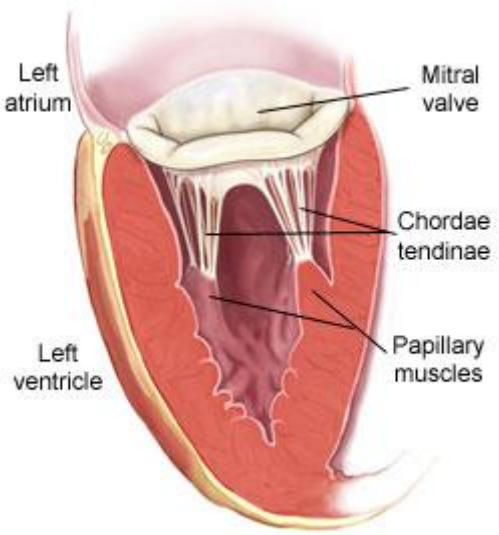

Ilustración 3. Sección transversal del ventrículo izquierdo y válvula mitral.

Este tipo de válvulas tienen un funcionamiento muy similar al de las válvulas semilunares. Una diferencia es que la parte inferior de estas está unida a los músculos papilares mediante varios tendones conocidos como cordón tendinoso. Cuando los ventrículos se contraen para bombear sangre al cuerpo, y cerrando así la válvula, los músculos papilares se tensan, y por extensión el cordón tendinoso, impidiendo que las solapas de la válvula se abra en sentido opuesto. (ilustración 3) [5].

Esto es necesario ya que los ventrículos generan mucha más presión cuando se contraen que las aurículas (la sangre tiene que recorrer más distancia comparado con llevarla de la aurícula al ventrículo) [3].

Estructura externa del corazón

Pasamos ahora a analizar la estructura externa del corazón, también imprescindible para su correcto funcionamiento.

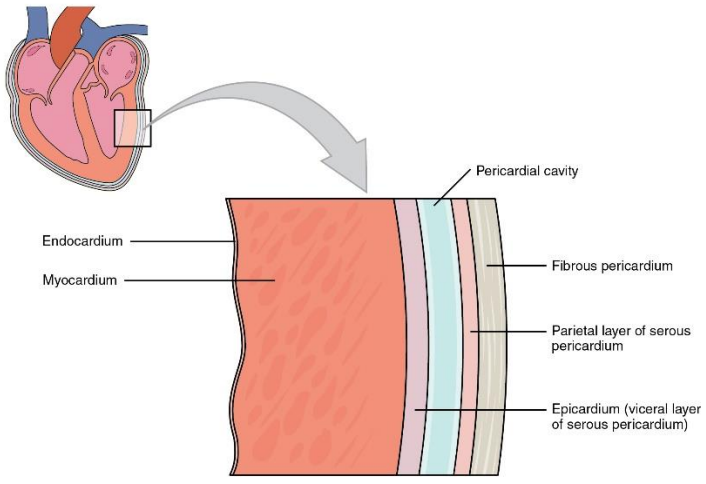

Ilustración 4. Membranas pericárdicas y capas de la pared del corazón

-Pericardio

El pericardio es una membrana de aproximadamente 2mm de grosor [6] que envuelve al corazón. Él cual está formado por dos sub bolsas, el pericardio fibroso y el pericardio seroso [7].

El pericardio fibroso, la capa externa, se encarga de proteger y mantener al corazón en el tórax, mientras que el pericardio seroso se encarga de lubricar el corazón, reduciendo la fricción entre el corazón y el resto de órganos en la caja torácica.

Esto es posible ya que el pericardio seroso a su vez está compuesto por 2 subcapas. El pericardio parietal, fusionado con el pericardio fibroso, y el pericardio visceral (o epicardio) que esta fusionado con el corazón (ilustración 4), dejando una cavidad entre ambas con líquido viscoso [3].

-Paredes del Corazón

Por paredes del corazón se entiende como las capas que componen el corazón, en contraste con el pericardio, que, como se ha explicado, envuelve al corazón pero no se considera parte del mismo.

Existen 3 capas que forman la pared del corazón, que son, del interior al exterior, el endocardio, miocardio y epicardio. Este último también forma parte de las paredes del corazón al estar fusionado con el corazón y el pericardio.

-Endocardio

El endocardio es una capa fina que está en contacto directo con la sangre de las cámaras cardiacas. Es una superficie lisa y no adherente, facilitando así la circulación de la sangre por las cámaras, además de elástica, algo imprescindible cuando el corazón se contrae [8].

-Miocardio

El miocardio, que significa literalmente músculo del corazón, es el músculo que hace posible que la sangre sea bombeada por todo el cuerpo. Rodea las cámaras del corazón y al contraerse prensa las cámaras del corazón, evacuando la sangre en su interior. Está formado por un tipo de células musculares especializadas denominadas cardiomiocitos [9].

Una particularidad única al miocardio es que estos cardiomiocitos están unidos mediante discos intercalados, que entre otras cosas permiten el paso de iones entre células por uniones gap, estas presentes no solo en el miocardio [10].

Al igual que el resto de músculos su contracción es posible debido a una diferencia de potencial de origen químico-orgánico entre las partes intracelulares y extracelulares [11], y a cambios bruscos en esta, conocido como potenciales de acción [12]. Como ocurre y es posible esta estimulación se explica en profundidad más adelante.

Según su función existen dos tipos distintos de cardiomiocitos en el miocardio [13].

- Células marcapaso: capaces de generar por si solas potencial de acción de forma periódica. Constituyen el 1% de los cardiomiocitos.
- Células contráctiles: se encargan de contraerse, haciendo que fluya la sangre, y transmiten los impulsos eléctricos a las otras células. Constituyen el 99% de los cardiomiocitos.

Aunque ambos tipos de célula se consideran técnicamente cardiomiocitos en adelante se usará este término para referirse solo a las células contráctiles.

-Epicardio

Otra de las características de esta capa, aparte de las ya mencionadas, es la de ser donde se encuentran las arterias coronarias (ilustración 2, C*oronary artery)*, encargadas de suministrar sangre a los músculos del corazón, el miocardio y músculos papilares [14]. Más específicamente se originan arriba de la válvula aórtica y van alejándose de esta y dividiéndose hasta que finalmente penetran hasta el miocardio [15].

Debido a la importancia del corazón y al trabajo constante que realiza, una correcta circulación de la sangre por este (circulación coronaria) es de vital importancia para aportar los nutrientes necesarios al miocardio y permitir su correcto funcionamiento. Si esta circulación se ve comprometida da lugar a problemas de salud graves, entre ellos, ataques al corazón [16].

Otra característica importante de estas arterias es la existencia en las arterias de anastomosis potencial. A diferencia de anastomosis verdadera relativamente común en los otros sistemas circulatorios.

Se dice que ocurre una anastomosis cuando dos arterias distintas o sus ramas se unen, dando lugar así a una especie de ruta alternativa en el caso de que se obstruya una de las arterias y evitar así la muerte del tejido al que irriga.

Pero, en el caso del corazón, aunque existen algunas anastomosis estas no ocurren entre arterias que irrigan el mismo tejido del miocardio, sino entre arterias que irrigan cada una distintas partes del miocardio. Por lo que si una arteria se bloquea, debido, por ejemplo, a una acumulación de colesterol, no habrá suficiente sangre causando la muerte del tejido y los graves problemas de salud que le acontecen. Sin embargo, si la obstrucción de la arteria se produce muy lentamente, se pueden desarrollar nuevos vasos sanguíneos y anastomosis [14] [4].

3.1.3 Sistema Eléctrico del Corazón

Anteriormente se ha visto como el corazón es una bomba que transporta sangre a todo el cuerpo y cómo las partes que lo componen hacen posible que funcione de forma eficaz, por ejemplo, sus válvulas para evitar el reflujo de la sangre o sus cámaras para evitar la mezcla de sangre oxigenada con la desoxigenada.

Sin embargo, para que todo esto sea posible el miocardio tiene que poder contraerse y relajarse de forma periódica, esto como el resto de músculos del cuerpo humano se origina debido a una estimulación eléctrica [17].

El sistema eléctrico del corazón es, por lo tanto, el responsable de generar estos impulsos periódicamente y de forma autónoma, así como de transmitirlos por todo el miocardio.

Conceptos previos a la creación de potencial eléctrico

La capacidad del cuerpo humano de generar corrientes eléctricas se basa en la presencia de iones en el fluido extracelular (líquido que rodea a las células) y en todas las células, entre ellos calcio (Ca^{++}), potasio (K^{+}), sodio (Na^{+}) y cloruro (Cl^{-}) [5].

Estos iones son fundamentales para el correcto funcionamiento del cuerpo humano. Como es lógico, dependiendo del proceso se necesitaran distintos tipos de moléculas y en mayor o menor cantidad [18], esto explica el desequilibrio en la concentración de los iones (tabla 1).

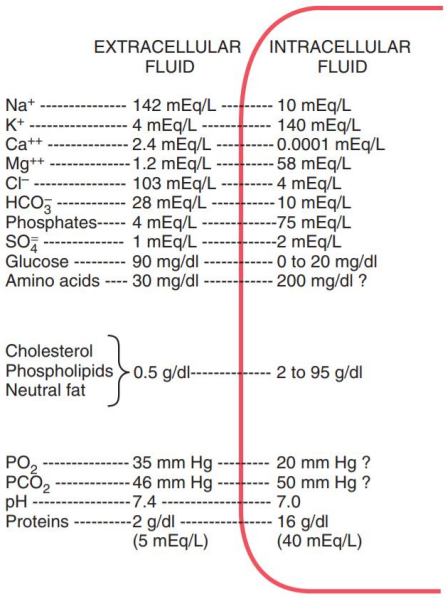

Tabla 1. Composición química de los fluidos extracelulares (izq.) e intracelulares (der). Medidos en miliequivalentes/litro. La línea roja representa la membrana celular.

Además, estos iones, o bien porque son usados en algún proceso metabólico, o bien porque son usados para transmitir señales a diferentes partes de la célula, necesitan de mecanismos que permitan el paso de iones entre las células y el fluido extracelular. Más adelante se ven en profundidad algunos de estos mecanismos.

-<u>Potencial de membrana</u>

El término potencial de membrana se refiere al potencial eléctrico que existe entre el interior de la membrana celular (membrana que separa el interior de la célula del exterior) y el fluido extracelular. Se considera el fluido extracelular como "toma de tierra" por lo que cuando el voltaje en el interior es más negativo que el exterior el potencial eléctrico también será negativo. Cuando el valor del potencial de membrana disminuye se conoce como hiperpolarización de la célula y despolarización en el caso contrario [19].

-<u>Transporte transmembrana de iones</u>

Fisiológicamente los mecanismos que permiten el flujo transmembrana de sustancias se pueden clasificar en dos tipos: por difusión o por transporte activo [5].

- Difusión: se basa en la tendencia estadística de las partículas a redistribuirse desde regiones de alta a baja concentración y conlleva a una distribución homogénea de las partículas [19].
- Transporte activo: se trata del movimiento de sustancias de regiones de baja a alta concentración. Puesto que este proceso ocurre "a contracorriente" se necesita de una fuente de energía adicional.

Sin embargo, tanto el fluido extracelular como el intracelular son eléctricamente neutros y si pudieran pasar libremente a través de la membrana celular no habría potencial de membrana. Esto no ocurre porque los caminos solo permiten el paso de ciertos tipos de iones.

Cuando estos iones son transportados, por ejemplo iones de sodio (Na^+) del fluido intracelular al extracelular (ilustración 5), estos intentaran llegar a un equilibrio. No obstante, conforme estos iones van entrando en el fluido extracelular este comenzará a ganar carga positiva y en el caso del fluido intracelular carga negativa. Mientras que la propia energía cinética de estas partículas las empuja a distribuirse uniformemente entre ambos fluidos el potencial de membrana ejerce una fuerza eléctrica opuesta, que irá aumentando gradualmente hasta que se contrarresten cesando así el flujo de iones. Este valor se conoce como potencial de difusión y puede ser calculado matemáticamente con la ecuación de Nernst cuando la célula solo es permeable a un ion y la ecuación de Goldman cuando lo es para más de uno [5].

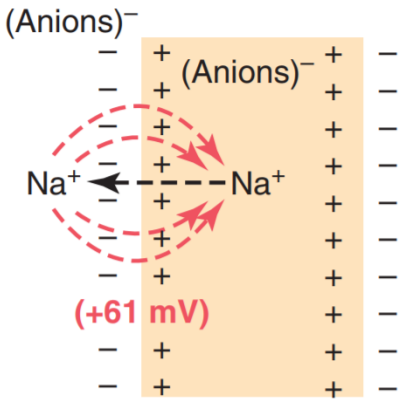

Ilustración 5. Potencial de difusión para una célula permeable al sodio.

Existen, a grandes rasgos, dos tipos de caminos en las células que son [5]:

- Bombas iónicas: encargadas del paso de iones por transporte activo [20].
- Canales iónicos (ilustración 6): son pequeñas aperturas proteicas en la membrana celular que permiten el paso, generalmente selectivo, de iones por difusión.

Además, pueden modificar su permeabilidad alternando entre dos estados: abierto y cerrado. Su comportamiento puede ser modelado con un sistema de "compuertas" en uno o ambos extremos del canal. Cuando ambas están abiertas el canal es permeable y cuando alguna está cerrada es impermeable. Cuando un canal iónico se abre o cierra en respuesta al potencial de membrana se denominan canales iónicos voltaje-dependientes [5].

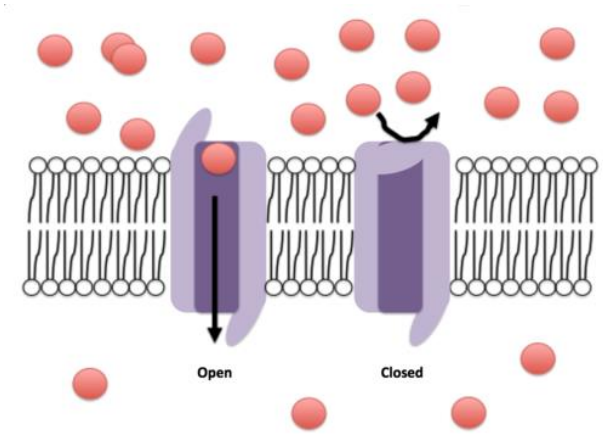

Ilustración 6. Canales iónicos en dos de sus posibles estados. Abierto (izq.) y cerrado (der.)

Puesto que los principales canales iónicos que caracterizan el ciclo cardiaco son todos voltaje-dependientes se usarán los términos canal y canal iónico voltaje-dependiente de forma intercambiable salvo si se especifica lo contrario.

Los canales iónicos voltaje-dependientes que tengan dos compuertas además pueden alternar entre estos dos estados mediante [5] [21]:

- Activación/desactivación: cuando se pasa de cerrado/abierto al estado opuesto debido a despolarización/hiperpolarización de la célula. Volviéndose permeable o no permeable respectivamente.
- Inactivación: Ocurre también por la despolarización de la célula, pero con un retardo temporal respecto a la activación, volviéndose no permeable. En este estado el canal no puede activarse independientemente del voltaje que posea la célula, este periodo se conoce como periodo refractario.
- Reactivación: Este proceso no ocurre hasta que la célula se hiperpolariza completamente y significa que la célula puede volver a activarse.

Los valores del potencial de membrana que causan estos procesos dependen del tipo de canal iónico voltaje-dependiente. El valor de potencial de membrana mínimo que causa la activación de un canal se denomina potencial de umbral.

Potenciales de acción cardiaco

El potencial de acción cardiaco es una subida seguida de una bajada brusca en el potencial de membrana de las células cardiacas. Es la causa por la que el corazón se contrae y relaja de forma cíclica.

A continuación se describe este proceso en profundidad para los cardiomiocitos y las células marcapasos de las aurículas.

Tomamos como el estado inicial cuando el corazón está relajado y listo para su próxima contracción.

-Potencial de acción en los cardiomiocitos

-Fase 0

Justo antes a esta fase (fase 4) los cardiomiocitos están relajados, es decir, no contraídos y poseen un potencial de membrana de $\backsim$-90mV. Esto se conoce como potencial de reposo.

Esta fase comienza cuando llega un estímulo eléctrico a la célula de alguna de sus células vecinas (ya sea otro cardiomiocito o una célula marcapasos) capaz de despolarizar la célula como mínimo a los $\backsim$-70mV [22].

Entonces se activan los canales rápidos de sodio (llamados así porque se activan en el orden de 0.1 ms) [5]. Debido a que la proporción de sodio (Na^+) es mucho más alta en el fluido extracelular (tabla 1) la célula se despolariza de forma casi inmediata (ilustración 7) hasta que llega a los $\backsim$+20mV.

Este potencial de membrana es mayor que el potencial de umbral de todos los canales iónicos que se irán activando a lo largo del resto de fases.

Estos canales son, entre otros, canales de calcio tipo-T y tipo-L.

- Canales tipo-T: tienen un potencial de umbral de $\backsim$-55mV [23]. Se abren en esta fase pero influyen relativamente poco a la despolarización de la célula [24].
- Canales tipo-L: tienen un potencial de umbral de $\backsim$-40mV [5]. Estos canales una vez abiertos tardan relativamente mucho más tiempo (varias décimas de segundo) en desactivarse que otros tipos de canales, también tardan más en abrirse, por esto los canales no influyen al potencial de membrana hasta la fase 2. [22]

Aunque se llamen canales de calcio estos son también permeables al potasio (K^+), pero esto no es muy importante porque estos canales son cientos de veces más permeables al calcio que al potasio. [5].

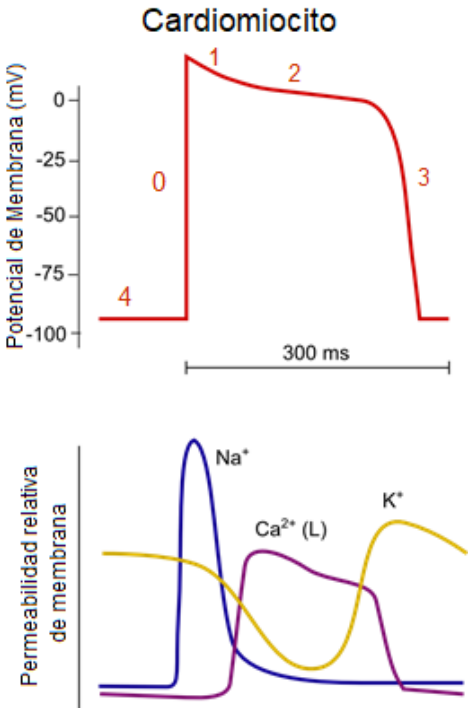

Ilustración 7. Potencial de membrana a lo largo de un potencial de acción (arriba) y cambios en la permeabilidad de la membrana a diferentes iones (abajo).

-*Fase 1* [22]

- Se inactivan los canales rápidos de sodio y desactivan los de calcio tipo-T.
- Se activan y rápidamente se inactivan canales de potasio (K^+). Esto causa que la célula se hiperpolarice poco pero rápidamente.

-*Fase 2* [22] [24] [25]

En esta fase el potencial de la célula se mantiene relativamente estable. Esto se debe a varios factores:

- Se abren un tipo de canales de potasio que son más permeables al paso de iones hacia dentro que hacia fuera. Estos canales siguen hiperpolarizando la célula pero no tanto como en la fase anterior. (hiperpolarización).
- Se activan finalmente los canales de calcio tipo-L (repolarización).
- La llegada abundante de calcio aumenta la actividad de las bombas iónicas sodio-calcio que por cada ion de calcio que exporta 3 de sodio son importados. (repolarización)
- Esta entrada de sodio aumenta la actividad de las bombas sodio-potasio que exporta sodio e importa potasio en un ratio de 3:2. (hiperpolarización).

-*Fase 3* [22]

En esta fase la célula se hiperpolariza de forma brusca y llegando hasta su potencial de reposo ($\backsim$-90mV).

- Se desactivan los canales de calcio tipo-L.
- Los canales de potasio que se abrieron en la fase 2 siguen abiertos.

Durante esta fase conforme va disminuyendo el potencial de membrana se van desactivando y/o reactivando los distintos canales, volviendo a estar listos para el próximo potencial de acción.

-*Fase 4* [22]

En esta fase el potencial de acción ya ha acabado y la célula está en su potencial de reposo. Los únicos canales que quedan abiertos son un subgrupo escaso de canales de potasio y varias bombas iónicas, algunas ya mencionadas en la fase 3, que siempre están funcionando.

Las bombas iónicas se encargan de mantener la concentración de iones estable que junto a los canales de potasio abiertos hacen que el potencial de difusión, y por consiguiente el potencial de reposo, sea ∽-90mV.

<u>-Potencial de acción en las células marcapasos</u>

Los mecanismos que causan potencial de acción en este tipo de células son similares al de los cardiomiocitos, sin embargo, su diferencia principal es que son capaces de iniciar potenciales de acción por si solos. Aun así, igual que los cardiomiocitos si reciben un estímulo eléctrico suficientemente alto (∽-40mV) también causará el inicio de un potencial de acción.

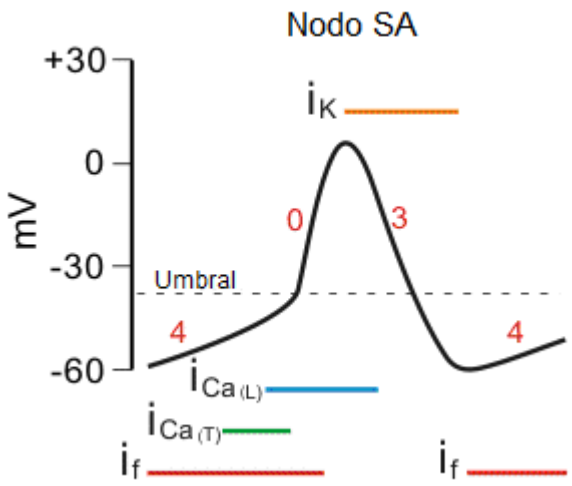

Ilustración 8. Potencial de membrana a lo largo de un potencial de acción. Mostrados también los momentos durante los que influyen los principales canales iónicos involucrados.

Se distinguen 3 fases análogas a la de los cardiomiocitos:

-*Fase 4* [26] [27] [28]

En esta fase estás células poseen un potencial de membrana de ∽-60mV este voltaje activa un tipo de canal iónico voltaje-dependiente llamados canales HCN relativamente poco permeables a cationes (I_f en ilustración 8).

Esto causa una despolarización gradual de la célula hasta llegar a los ∽-55mV cuando se activan los canales de calcio tipo-T que despolarizan aún más la célula hasta los ∽-40mV. [23] [26].

-Fase 0

Esta fase comienza con la activación de los canales de calcio tipo-L, despolarizando así la célula de forma brusca hasta ∽+40mV. Durante esta fase los canales HCN y de calcio tipo T se van inactivando y pasados unos 100-150 milisegundos [5] también lo hacen los canales de calcio tipo-L finalizando esta fase.

La razón por la que los canales rápidos de sodio no se activan en esta fase a pesar de haber sobrepasado su potencial de umbral es que no existen en este tipo de células [29].

-Fase 3 [30]

Esta fase comienza con la desactivación de los canales de calcio tipo-L. El cambio de potencial de membrana de negativo a positivo activa canales de potasio que van hiperpolarizando la célula. Estos se van inactivando gradualmente hasta que se vuelve al potencial de reposo de ∽-60mV y vuelve a comenzar la fase 4.

Una vez más las bombas iónicas se encargar de mantener la concentración de iones a los niveles iniciales para permitir que pueda volver a generarse otro potencial de acción.

Anatomía y fisiología del sistema eléctrico del corazón

Una vez visto como las células del corazón son capaces de generar y transmitir electricidad de forma periódica se va a explicar cómo estos mecanismos se sincronizan e interactúan entre ellos para hacer que el corazón bombee correctamente.

-Nodo Sinoatrial

Todo empieza en el nodo sinoatrial (Nodo SA) situado en el miocardio de la aurícula derecha cerca de la vena cava superior, compuesto por células marcapasos y del tamaño de unos pocos centímetros [4]. Este nodo es capaz de generar potenciales de acción unas 60-100 veces por minuto. Cada potencial de acción corresponde a una pulsación del corazón por lo que este nodo es el encargado de regular el ritmo cardiaco [31].

Sin embargo, la frecuencia a la que se generan estos potenciales de acción puede ser influenciada por el sistema nervioso simpático y el parasimpático. El primero es capaz de aumentar la frecuencia mediante una serie de reacciones que crean un tipo de moléculas que hacen que los canales HCN sean más permeables esto hace que se inicien sus potenciales de acción más rápidamente.

La función del sistema nervioso parasimpático es el opuesto, disminuir la frecuencia cardiaca, es capaz de reducir la actividad del sistema nervioso simpático además de activar más canales de potasio reduciendo el potencial de membrana. Ambos sistemas son necesarios, por ejemplo, para aumentar el ritmo cardiaco cuando realizamos ejercicio o disminuirlo cuando dormimos [31].

-Vías Internodales

Los potenciales de acción generados por el nodo sinoatrial van transmitiéndose a las células vecinas gracias a las uniones gap hasta llegar al nodo atrioventricular (ver más adelante). Conforme van generando potenciales de acción también se contraen. Sin embargo, se puede observar como los impulsos eléctricos se transmiten unas 3 veces más rápido de forma paralela a las fibras de cardiomiocitos que perpendicularmente [4].

Para explicar este fenómeno se conjeturó que existían una serie de caminos especializados de menor resistencia eléctrica. Actualmente el consenso científico es que la existencia de estas vías no se debe a algún tipo de células especiales sino como un resultado debido a la estructura del miocardio. Es decir, estas vías solo existen desde un punto de vista morfológico y no anatómico [4] [32].

-Nodo Atrioventricular

Este nodo (Nodo AV), más pequeño que el sinoatrial, está formado por células marcapasos que generan potenciales de acción alrededor de 40-60 veces por segundo [5]. Esto hace que no generen sus potenciales de acción de forma natural ya que les llegan antes los generados por el nodo sinoatrial.

Una característica muy importante de este nodo es que actúa también de retardador de las señales eléctricas. Debido al bajo número de uniones gap en este nodo las señales eléctricas tardan $\backsim 0.09$s en llegar hasta las células cardiacas de los ventrículos. [5] Este retardo hace posible que las aurículas se contraigan completamente primero y pase la sangre a los ventrículos antes de que estos se contraigan.

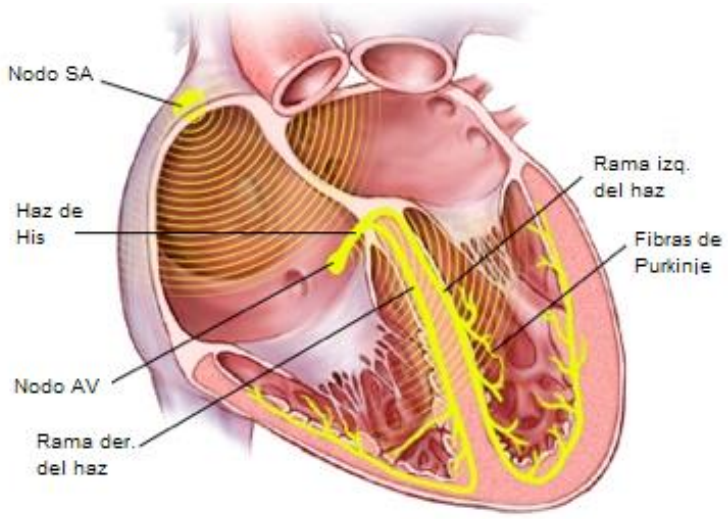

Ilustración 9. Componentes del sistema eléctrico del corazón.

-Haz de His

El haz de His es la conexión eléctrica entre las aurículas y ventrículos que de otra forma estarían aisladas eléctricamente por el tabique atrioventricular. De esta forma los cardiomiocitos de los ventrículos solo pueden ser excitados por el haz de His, cardiomiocitos ventriculares vecinos o fibras de Purkinje (ver más adelante) y no por el potencial de acción de los cardiomiocitos auriculares.

Este nodo posee una primera sección, la que penetra el tabique atrioventricular, que no conduce la electricidad especialmente bien. Esta introduce un retardo adicional de $\backsim 0.03$s en la propagación de la electricidad por el corazón [5].

Tras penetrar el tabique se originan dos ramas (rama izq. y der. del haz) que recorren la zona ventricular en dirección a la base del corazón hasta que finalizan y dan paso a las fibras de Purkinje.

-Fibras de Purkinje

Estas fibras son las encargadas de transmitir los potenciales eléctricos a los cardiomiocitos de los ventrículos de forma casi simultánea para que se contraigan a la vez generando así la presión extra que necesitan estas cámaras. Esto se consigue por un lado debido a su estructura arbolea (ilustración 9) que distribuye los potenciales eléctricos a varios puntos de las cámaras y por otro a su alta velocidad de transmisión de la electricidad.

Estas fibras, de mayor tamaño que las del miocardio ventricular, transmiten la electricidad a una velocidad de 1.5 a 4 m/s (6 veces más rápido que de cardiomiocito a cardiomiocito). Esto parece que se debe a que las uniones gap que las unen son relativamente mucho más permeables a los iones. [5].

También las fibras de Purkinje están formadas por células marcapasos capaces de generar potenciales de acción 15-40 veces por segundo, [5] pero, al igual que en el nodo atrioventricular son “anulados” por los del nodo sinoatrial.

3.2 Simulación Cardiaca

En la sección anterior se ha visto cómo funciona el corazón, esto es, obviamente, un prerrequisito imprescindible para poder realizar simulaciones por ordenador del corazón.

Los programas que se usan en la simulación cardiaca implementan modelos matemáticos que reproducen el comportamiento del corazón, estos modelos están basados en nuestros conocimientos actuales del corazón y, por lo tanto, van cambiando y mejorándose conforme se van haciendo nuevos descubrimientos científicos.

Por lo general la simulación de procesos biológicos y específicamente la simulación del corazón presentan algunas peculiaridades que complican la creación de modelos realistas. Por ejemplo, un problema es la dificultad poder observar el corazón mediante métodos tradicionales. Al estar situado dentro del cuerpo humano es imposible verlo y tomar medidas sin cirugía invasiva, además está el problema de que cualquier dificultad o error que ocurra puede causar la muerte. Sin embargo, hay varias formas de solventar este problema, uno de ellos es mediante electrocardiogramas donde se ponen varios electrodos en la piel para medir la actividad eléctrica del corazón [33] o mediante imágenes por resonancia magnética capaces de ver el órgano por dentro sin necesidad de cortarlo ni cirugía [34].

3.2.1 Modelado del corazón

El primer paso para poder realizar simulaciones cardiacas es el de obtener un modelo geométrico 3D del corazón.

El primer modelo computarizado del corazón fue desarrollado por Okajima et al. (1968) (ilustración 10) [35] (aunque se excluyeron las aurículas) donde se obtuvo el corazón de un cadáver y tras ser puesto en una solución gelatinosa y congelado se cortaron 30 secciones horizontalmente de 3mm de grosor cada una. Estas secciones fueron fotografiadas frente a una cuadricula de 30x30 de 3mm. Tras pasar las coordenadas manualmente al ordenador se obtuvo finalmente un modelo del corazón circunscrito en una matriz de 27,000 voxels.

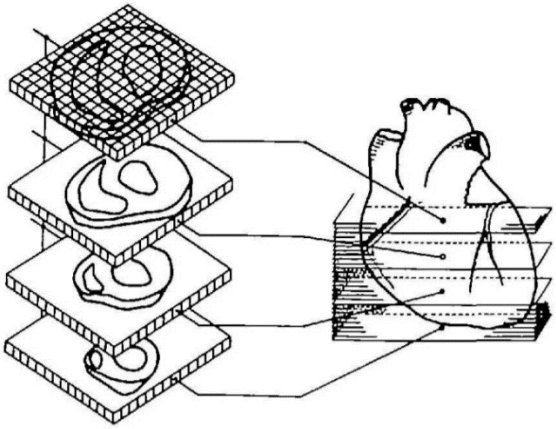

*Ilustración 10. Método de modelado y digitalización empleado por Okajima et al **[35]**.*

No fue hasta 1993, gracias a una tecnología recientemente desarrollada: el escáner TAC, cuando Lorange et al. obtuvieron 132 secciones de 1mm de grosor a una resolución de 512x512 píxeles. Esto produjo eventualmente un modelo compuesto por una matriz de 250,000 voxels [36] [37].

Con el paso de los años y las consiguientes mejoras tecnológicas se han podido obtener modelos aún más precisos. Por ejemplo, gracias a The Visible Human Project (El proyecto humano visible, en español), llevado a cabo por la Biblioteca Nacional de Medicina de los Estados Unidos, se obtuvieron, entre otros, fotografías de 1,871 secciones de 1mm de grosor a una resolución de 2048x1216 píxeles [38]. Esto permitió la creación de modelos cardiacos alrededor de los 450,000 elementos [39] [40].

Adicionalmente, ha habido modelos aún más precisos como el creado por Hren y Horacek (1997, 1998) [41] llegando hasta el 1,700,000 de elementos.

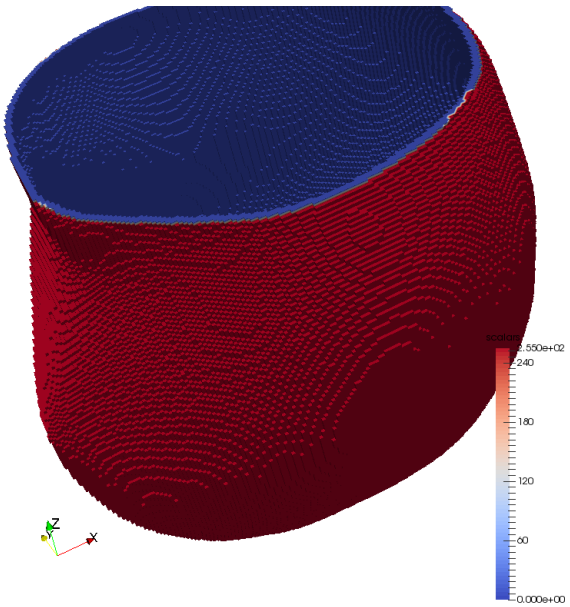

Ilustración 11. Uno de los modelo ventriculares usado actualmente en la UV. Compuesto por varios millones de elementos.

Modelado de fibras de Purkinje

Como se ha visto el principal método por el que se ha modelado históricamente el corazón ha sido mediante imágenes de secciones transversales del corazón, sin embargo, en estas es imposible visualizar la red de fibras de Purkinje en el cuerpo humano [36] debido a su estructura microscópica.

Sin embargo, sí ha habido otros experimentos con éxito que han conseguido visualizar estas fibras mediante procesos inmunohistoquímicos aunque estos han sido realizados solo en corazones no humanos como en corazones porcinos por Toshimori H. et al. (1988) [42] o por Aneel Ansari et al. (1999) en corazones de oveja [43].

Debido a esto, la forma usual de modelar estas fibras se realiza afrontando el problema desde un punto de vista inverso. Se realizan experimentos donde se mapean en que instante se excitan las células cardiacas y en base a esto se modelan las fibras de Purkinje para que se ajusten a estos datos experimentales [36].

Uno de los estudios más importantes en este campo fue llevado a cabo por Durrer et al. (1970) (ilustración 12) [44] donde se pusieron hasta 870 electrodos en cada uno de los 7 corazones humanos usados.

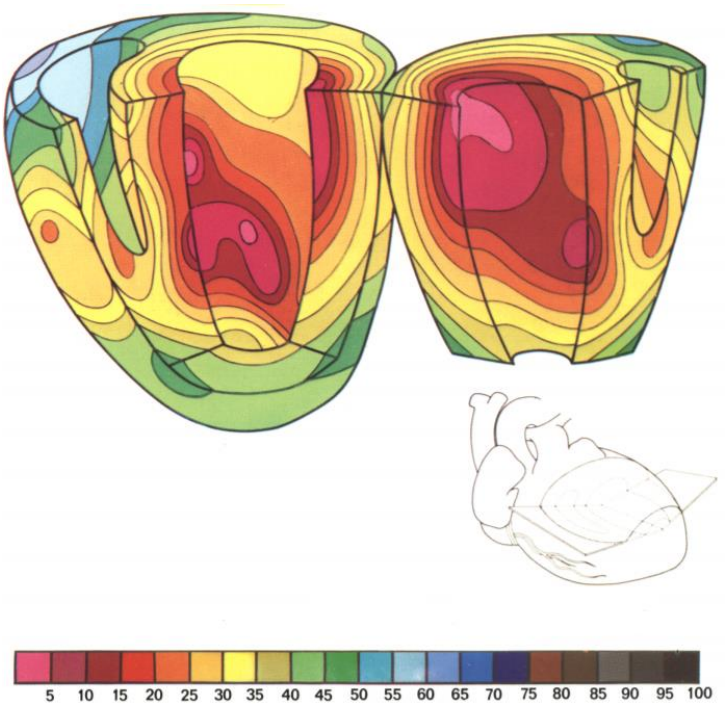

Ilustración 12. Representación isocrónica en 3D de tiempos de activación del corazón obtenido por Durrer et al. (1970) [44].

Gracias a este estudio Aoki et al. (1987) [45] realizo un modelo 3D del corazón de unas 50,000 células (resolución espacial de 1.5mm) que modelaba también partes del sistema

eléctrico del corazón como el haz de His y fibras de Purkinje. O el artículo [37] ya mencionado que incluía también el sistema eléctrico e incluía anisotropía eléctrica.

Finalmente destacar el artículo de Simelious et al. (2001) [46], que usaba como base el modelo del corazón de Hren et al. [41]. Este modelo es importante porque modela el sistema eléctrico del corazón desde un punto de vista anatómico y no solo en base a datos experimentales como en los anteriores casos. Primero crea un modelo inicial del sistema eléctrico (ilustración 13) para luego ser ajustado en base a los datos experimentales que se intentan replicar. Para replicar los datos obtenidos en electrocardiogramas o vectocardiogramas se ajustaron los tiempos de activación entre ventrículos y para los de Durrer et al. [44] la posición y número de las uniones entre fibras de Purkinje y los cardiomiocitos. Sin embargo, no se pudo crear un modelo unificado que satisficiera los 3 conjuntos de datos.

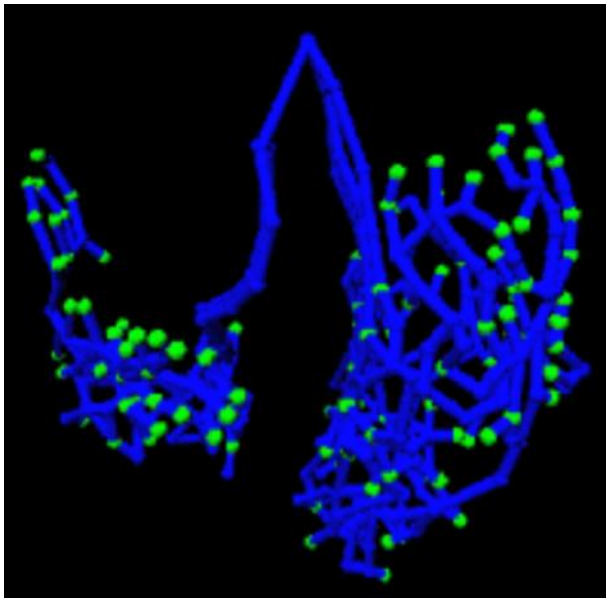

Ilustración 13. Geometría inicial del sistema de conducción empleado por Simelious et al. (2001). Los puntos verdes son puntos de unión entre fibras de Purkinje y células cardiacas [46].

3.2.2 Modelos matemáticos

Con la obtención de modelos realistas del corazón y su sistema eléctrico se hace posible realizar simulaciones por ordenador del corazón.

Sin embargo, las ecuaciones en derivadas parciales que describen la conducción eléctrica en el campo de la bioelectricidad son complejas y solo se pueden calcular en un tiempo de computación aceptable mediante métodos de aproximación numérica. Aunque usar modelos del corazón más simples reduciría el tiempo de cálculo, sin la ayuda de estos métodos se tendrían que simplificar demasiado añadiendo demasiado error [47].

De los diversos métodos de aproximación numérica que existen dos de los más usados en el campo de la simulación cardiaca son: método de los elementos finitos y método de las diferencias finitas [48].

A grandes rasgos estos métodos se basan en los mismos conceptos. Para geometrías irregulares las ecuaciones que permiten el cálculo de conducción eléctrica son demasiado complejos. Estos dividen la geometría en formas sencillas como tetraedros que permiten usar ecuaciones más simples sin añadir demasiado error.

Método de las diferencias finitas [47]

1. Se divide la geometría en elementos geométricos simples y uniformes.
2. Para cada nodo se calcula una ecuación sustituyendo las derivadas por la aproximación:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

   Siendo h la distancia entre nodos.

3. Se obtiene una matriz combinando la ecuación recientemente obtenida de cada nodo.
4. Tras incluir los valores de frontera se resuelve la matriz.

Método de los elementos finitos [47] [49]

1. Se divide la geometría en varios elementos geométricos simples, en este caso suponemos una geometría en 2D.
2. Cada uno de los nodos de los elementos se describen mediante ecuaciones algebraicas más sencillas que referencian nodos adyacentes.
3. Se obtiene de cada nodo un sistema de ecuaciones evaluando su ecuación, obtenida del paso 2, con cada uno de los nodos.
4. Se combinan los sistemas de ecuaciones en una sola matriz. Esto es posible porque cada dos elementos vecinos tendrán nodos comunes que aparecerán en ambos sistemas de ecuaciones.
5. Se sustituyen en la matriz los valores de frontera. Esto hace que el sistema de ecuaciones pase a ser compatible determinado.
6. Se calcula la matriz inversa obteniendo la solución para los nodos. También se puede obtener una solución continua interpolando los valores entre nodos.

3.3 Estudio del *Framework*

En este apartado se van a analizar las herramientas y programas informáticos relevantes para hacer posible el desarrollo del proyecto. Se verán aspectos como historia del proyecto, bajo que licencia están publicados, módulos que tienen, etc. Todo esto con el objetivo de adquirir un conocimiento previo al diseño de la aplicación y las posibles limitaciones o dificultades que haya que superar.

3.3.1 The Visualization ToolKit (VTK)

The Visualization ToolKit (El Kit de Herramientas de Visualización, en español) es un software de sistema de código abierto para el desarrollo de aplicaciones de procesado de imágenes, gráficos 3D y visualización de datos [50]. Actualmente se encuentra en su versión 8.1.1 lanzada el 14 de mayo de 2018 y es compatible con Windows, iOS, Linux, MacOS y como servicio Web [51].

El software está desarrollado en C++ debido a su alto rendimiento y basado en el paradigma de programación orientado a objetos. Sin embargo, desde 1998 soporta diversos lenguajes de programación. A día de hoy soporta Java, Python, .NET y Tcl [52] aunque mediante el uso de *wrappers.*

Desarrollado originalmente en 1993 por 3 investigadores. Con el paso de los años y gracias a su código abierto y modelo de colaboración público este fue ganando usuarios y contribuidores hasta que en 1998 parte de los desarrolladores fundaron Kitware para centrarse en el desarrollo y soporte del software. Actualmente posee más de 400 contribuidores [53] como la Biblioteca Nacional de Medicina de los Estados Unidos o el Laboratorio Nacional de Los Álamos [54].

VTK *pipeline* [55]

Debido a la alta complejidad y coste computacional que puede suponer el renderizado de entornos 3D donde, como se ha visto anteriormente, se trabaja con varios millones de elementos, es importante que los datos se traten de forma eficaz e intentar disminuir el número de operaciones sobre ellos, todo esto sin afectar las funcionalidades del programa.

Por esto conocer los procesos y el orden mediante los que se opera sobre los elementos es algo importante tanto para entender cómo funciona esta librería como para usarla. Estos procesos se conoces como la *pipeline* del programa (ilustración 14).

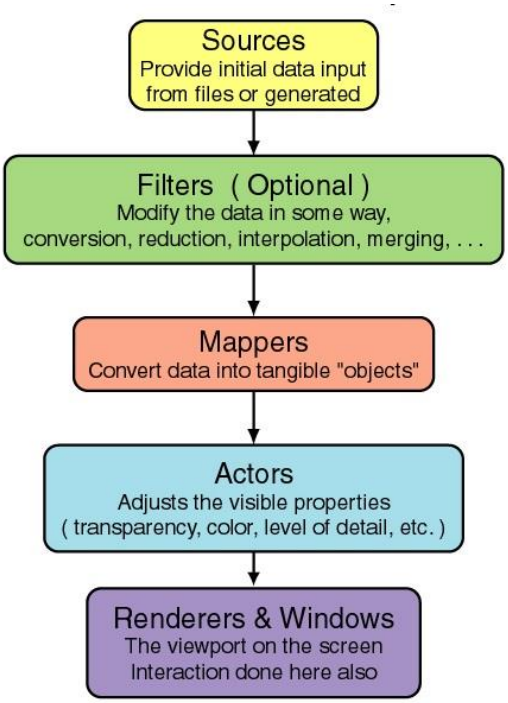

Ilustración 14. Pipeline *de la librería VTK.*

-Fuentes (*Sources)*

Primera fase de la *pipeline*, aquí se proporciona a la aplicación los datos de entrada iniciales de los elementos con los que se va a trabajar. VTK genera este tipo de datos mediante clases de las que se instancian objetos, por ejemplo existe la clase *vtkConeSource* que define geométricamente un cono en base a su altura, radio, centro y orientación.

-Filtros (Filters*)*

Modifican las fuentes, son conceptualmente similares, modifican las fuentes y dan como datos de salida fuentes. Son opcionales.

-Mapeadores (*Mappers)*

Transforma las fuentes en primitivas geométricas como triángulos o tetraedros. En el ejemplo anterior el cono pasa de estar definido por su altura, centro, etc. a ser definido por una serie de nodos y como están conectados entre ellos.

-Actores (*Actors)*

Modifican la apariencia sobre cómo se visualizaran los mapeadores en pantalla, aspectos como color o transparencia.

-Renderizadores y ventanas (*Renderers* & *Windows*)

Transforma la escena 3D en una imagen 2D que es la que visualiza el usuario. Cada renderizador crea una imagen en 2D que son visualizadas en una ventana.

Ficheros en VTK [56]

VTK usa varios tipos de ficheros para almacenar la información de los diferentes tipos de objetos que usa. De esta forma se pueden reutilizar por otros programas. Estos ficheros contienen principalmente dos tipos de información: unos guardan un conjunto de datos que definen toda la geometría de la escena y los otros además almacenan información adicional como actores, luces, cámaras, etc. por lo que pueden recrear una escena mientras que los primeros solo contienen información de las fuentes.

Para este proyecto interesa solo ser capaz de leer e interpretar el primer tipo de ficheros, VTK proporciona varias clases que leen este tipo de ficheros y los transforman en instancias de una clase que contiene toda la información del fichero. Son los que VTK denomina *readers* (lectores).

3.3.2 Cardiac Arrhythmia Research Package (CARP)

El Cardiac Arrhythmia Research Package (Paquete de investigación de arritmias cardiacas, en español) se trata de un simulador multipropósito bidominio [57]. Este modelo matemático describe las propiedades eléctricas del miocardio, teniendo en cuenta propiedades como anisotropía eléctrica en los espacios intracelulares y extracelulares [58].

CARP es un programa de código cerrado compatible solo con sistemas operativos basados en Linux [59].

CARP tiene varios módulos distintos que realizan tareas muy diversas desde simulación de células cardiacas hasta visualizadores de mallas. Los dos módulos que más interesan para este proyecto son: mesher y especialmente Tarantula.

Ambos son generadores de mallas, mientras que mesher es un módulo sencillo pensado para test rápidos, Tarantula al ser más complejo y tener muchas más funcionalidades es el módulo propio de CARP que se usa para generar la geometría del corazón.

Tarantula

Tarantula es un generador de mallas basado en un software para el modelado en aplicaciones industriales: Spider, producido por CAE Software Solution [59]. De acuerdo a la página web de CARP el módulo Tarantula soporta, entre otras, las siguientes funcionalidades [60]:

- Generar mallas desestructuradas, es decir, se necesita especificar explícitamente sus relaciones de vecindad.
- Crear mallas con distintas resoluciones espaciales, por ejemplo, se pueden modelar secciones del corazón más simples con un número menor de elementes mientras que las partes más delicadas o complejas pueden ser detalladas con más elementes, todo esto como una malla uniforme.
- Compatibles con el método de los elementos finitos, aunque actualmente solo soporta algunos elementos geométricos (ilustración 15).
- Basado en árboles octales: un tipo de estructura de datos arbolea donde cada nodo tiene 8 hijos.

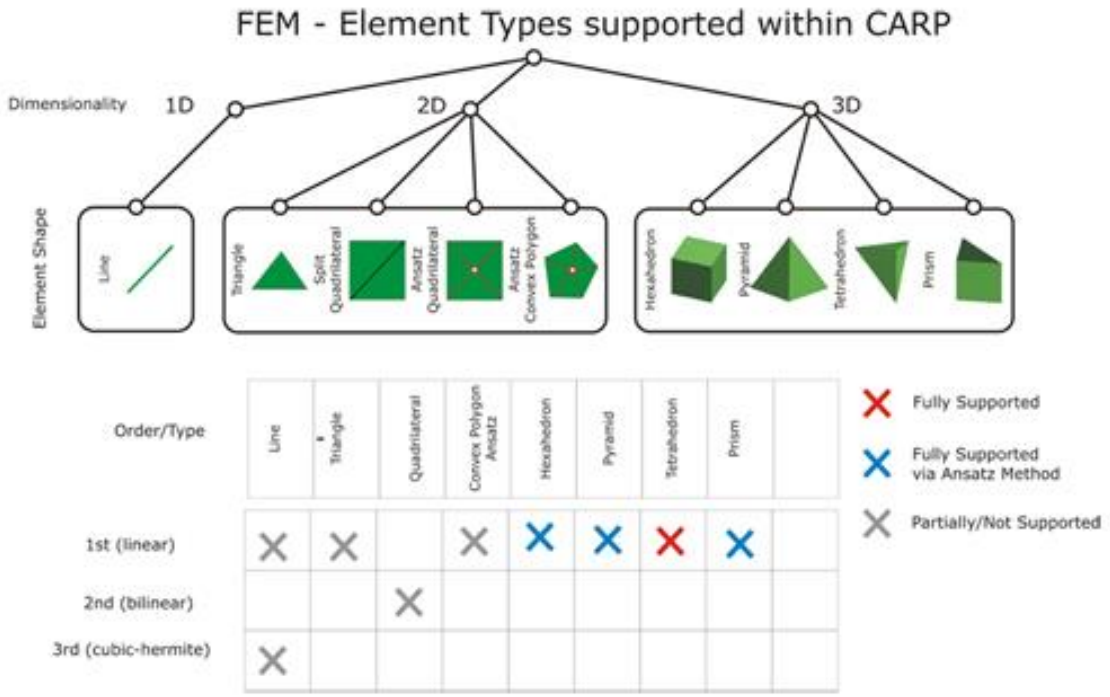

Ilustración 15. Primitivas para las que el método de los elementos finitos está soportado.

Ficheros que definen la geometría del corazón

Según el manual de usuario de CARP existen 3 ficheros distintos obligatorios que especifican el modelo del corazón a usar por el resto del programa. Estos son un fichero de nodos, un fichero de elementos y un último para la orientación de fibras. De estos ficheros los dos primeros son los que más interesan.

Fichero de nodos

La sintaxis del fichero de nodos es bastante sencillo, la primera línea es solamente el número total de nodos de la malla seguida de las coordenadas de cada nodo en µm. La extensión de este fichero es .pts. Por ejemplo:

10000

10.3400 -15.0023 12.0234

11.4608 -12.0237 11.9821

...

-8.4523 36.7472 4.6742

Fichero de elementos

Este fichero, con extensión .elem, es donde se le dice al programa los elementos que componen la malla del corazón y cuáles son sus nodos. CARP soporta el uso de varias figuras geométricas simultáneamente en una única malla, los tipos de elementos soportados se encuentran más abajo (tabla 3), mientras que la sintaxis del fichero, que es algo más complicada que para los nodos, puede verse en el anexo 1.

Type Specifier	Description	Interpolation	#Nodes
Ln	Line	linear	2
cH*	Line	cubic	2
Tr	Triangle	linear	3
Qd	Quadrilateral	Ansatz	4
Tt	Tetrahedron	linear	4
Py	Pyramid	Ansatz	5
Pr	Prism	Ansatz	6
Hx	Hexahedron	Ansatz	8

* not implemented

Tabla 2. Tipo de elementos soportados por CARP

Fichero que define fibras de Purkinje

De acuerdo al manual de usuario de CARP las fibras de Purkinje se definen en un fichero opcional con la extensión .pkje. La estructura exacta del fichero se puede encontrar en el anexo 2, sin embargo sus principales características son:

- Las fibras de Purkinje están definidas mediante cables (líneas unidimensionales) que a su vez están compuestos por segmentos (rectilíneas) que van de nodo a nodo.
- Los cables están 0-indexados, es decir, cada cable es identificado por un número que va desde el 0 hasta numero_de_cables-1.
- El cable 0 se considera el origen de las fibras de Purkinje por lo que representa el haz de His.
- Cada cable puede tener hasta 2 padres y/o 2 hijos.
- La definición de los cables tiene que estar ordenada de menor a mayor según su índice, empezando por el 0 y sin saltarse ninguno. Para un cable con 2 padres y/o 2 hijos el primero debe de ser aquel con el índice más bajo.
- Los cables sin hijos se asume que son un punto de unión con el miocardio.

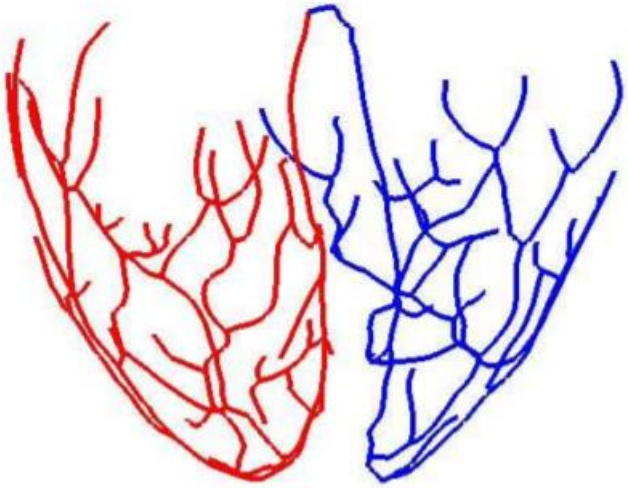

Ilustración 16. Visualización en CARP de un modelo 3D de fibras de Purkinje.

4 Búsqueda y Selección de Herramientas

Una vez recopilada información sobre el estado del arte del corazón y su simulación se va a proceder a analizar y seleccionar las herramientas y software que se usaran para el desarrollo de lo que resta del proyecto. Por lo general se tratará de usar software con el que ya este familiarizado, para facilitar desarrollo del trabajo, salvo en el caso de que se necesite alguna funcionalidad en particular.

4.1 Diseño de la Aplicación

Antes de desarrollar la aplicación es necesario diseñarla, en este caso se tienen que realizar los diseños en UML. Para esta tarea se ha decido usar el programa llamado Visual Paradigm por varios motivos:

- Compatible con los sistemas operativos Windows, Mac y basados en Linux [61].
- Único programa de diseño de software en el que se posee experiencia previa.
- Aun siendo un programa comercial, la Universidad de Valencia proporciona a los alumnos que están realizando el TFG una licencia académica gratuita.

4.2 Lenguaje de Programación

A grandes rasgos el software constará de 4 partes: lectura del fichero de entrada, parsear la malla, generar y poblar las estructuras de datos necesarias y escribir el fichero de salida. La lectura y escritura de ficheros son tareas de bajo nivel y lo contrario las otras partes.

Este proyecto será realizado en el lenguaje de programación C++ debido a:

- VTK es una librería nativa de C++, se evitan problemas de compatibilidad.
- Lenguaje del que se posee un conocimiento más en profundidad.
- Lenguaje de bajo nivel perfecto para poder realizar de forma óptima la lectura y escritura de los ficheros, algo importante si los ficheros poseen gran cantidad de información.
- Cuenta en su librería estándar una cantidad muy amplia de estructuras de datos ya implementadas y optimizadas que pueden ser muy útiles para la 3ª tarea anteriormente mencionada.

4.3 Sistema Operativo

El principal aspecto que hay que tener en cuenta a la hora de elegir el sistema operativo es que sea compatible con los programas que se van a usar. Tanto VTK como Visual Paradigm soportan los principales sistemas operativos, sin embargo, CARP solo soporta Linux. Por lo tanto se realizara el programa en Linux, más específicamente en Ubuntu 18.04.

4.4 Entorno de Desarrollo Integrado (EDI)

Aunque el uso de un EDI no es imprescindible su uso facilita mucho la tarea de programar, por ejemplo, corrección de errores ortográficos, autocompletar código etc. Debido a que

no se necesita ninguna funcionalidad específica la elección del EDI es meramente personal. Por esto se usará NetBeans; a destacar: soporta Linux, gratuito, y se tiene experiencia previa con él.

5 Planificación y Especificación

En este apartado habiendo adquirido ya un conocimiento conceptual sobre el trasfondo del proyecto así como las herramientas a usar durante el mismo se procede a su planificación. Se trata de estudiar que tareas debe de cumplir la aplicación, cómo se dividirán las tareas, que dificultades o problemas pueden ocurrir estando así preparados de antemano.

5.1 Análisis de Requisitos

Aquí se presentarán las tareas y funcionalidades que la aplicación debe de cumplir.

5.1.1 Requisitos funcionales

- Debe de poder convertir mallas de fibras de Purkinje modeladas en VTK a un fichero de fibras de Purkinje (.pkje) compatibles con CARP.
- Debe de poder convertir mallas del corazón modeladas en VTK a ficheros de nodos (.pts) y fichero de elementos (.elem) compatibles con CARP.

5.1.2 Requisitos no funcionales

- Código fácil de leer y mantener: tanto VTK como CARP está aún en desarrollo y puede que añadan nuevas funcionalidades o realicen cambios que obliguen a modificar el programa. Este proceso, seguramente, tendrá que ser realizado por otro programador, de ahí la importancia de este requisito.
- Generar documentación de la aplicación mediante Doxigen.

5.2 Especificaciones del Sistema

El sistema operativo deberá estar basado en Linux, aparte de esto debido a que es un programa intrínsecamente poco exigente se van a tomar como requisitos mínimos los necesarios para ejecutar CARP: 8GB de memoria RAM y un procesador de 8 núcleos.

5.3 Planificación

Antes de comenzar con la implementación del proyecto no solo basta con adquirir los conocimientos necesarios para el mismo, sino que también hay que saber si el proyecto es viable, es decir, si nos conviene realizarlo. Aspectos como el tiempo necesario para realizar el proyecto, su coste o la legalidad del mismo son muy importantes tanto para poder estar preparado a las dificultades del proyecto como poder realizarlo de forma eficaz.

5.3.1 Descomposición de tareas

El primer paso es el de enumerar cada una de las fases que forman el proyecto y las tareas que componen cada una de las fases. De esta forma se puede proceder a obtener una estimación de cuánto tiempo se tardará en hacer el proyecto y poder cumplir las fechas tope del proyecto o modificar algunas tareas para ajustarse a los plazos deseados. Las fases en cuestión son:

1. **Propuesta del proyecto:** El primer paso es el de definir el problema que se pretende solucionar así como redactar su anteproyecto donde se realiza un análisis preliminar sobre qué objetivos se desean cumplir al final del proyecto y del tiempo aproximado que se tardará en realizarlos.

2. **Estado del arte:** Se realiza un estudio en profundidad del conocimiento vigente que existe del tema sobre el que se basa el proyecto además de las técnicas, tecnologías, metodologías y sus avances empleados en proyectos similares o anteriores a lo largo del tiempo para poder afrontar el resto del proyecto con juicio. Conceptualmente se ha dividido en 3 partes: el corazón, donde se estudia el funcionamiento del corazón como base para afrontar el resto del proyecto; simulación cardiaca, como estos conocimientos se han aplicado en este campo y estudio del *framework*, como funcionan específicamente los programas en los que se basa la aplicación.

3. **Búsqueda de herramientas:** Se seleccionan las herramientas que se tienen que usar para el desarrollo del proyecto, o bien para cumplir los objetivos del proyecto, o para adaptarse a los estándares o métodos empleados por otros profesionales. Se trata de elegir y justificar el lenguaje de programación, sistema operativo y otros programas que se usarán a lo largo del proyecto.

4. **Planificación:** Como se ha visto se trata de especificar los requisitos que cumplirá la aplicación así como realizar una estimación de los costes temporales y económicos del proyecto así como posibles leyes que le afecten.

5. **Análisis y diseño:** Mediante el uso de diagramas se terminan de detallar los requisitos de la aplicación y se establece como se organizara el código. Se expondrán tanto los casos de uso de la aplicación como el diagrama de clases.

6. **Implementación:** en esta fase es donde se programa la aplicación, en primer lugar se el alumno debe de instalar las herramientas que va a usar así como familiarizarse con ellas. La aplicación podrá convertir mallas modeladas en VTK y convertirlas a ficheros compatibles en CARP tanto mallas de fibras de Purkinje como modelos del corazón.

7. **Pruebas:** tras realizar el programa se pasa a la realización de pruebas sobre este para poder asegurarse de que funciona correctamente, solucionando cualquier error que se encuentre y asegurándose de que cumple los requisitos de como de forma eficaz.

8. **Conclusiones:** Se analiza el proyecto ya finalizado, se ven posibles cambios o mejoras adicionales que se podrían añadir en retrospectiva y un resumen de los conocimientos y destrezas adquiridos a lo largo del proyecto.

5.3.2 Planificación temporal

Una vez identificadas las fases del proyecto se pasa a estimar cuanto tiempo se empleara en cada una de ellas y sus subtareas principales, sin embargo, esta estimación tiene que ser obviamente fundada y que sea una aproximación realista del tiempo que se tardará en hacerlas, de lo contrario no serviría de nada.

Para lograr estos objetivos existen ciertas técnicas como el juicio de expertos. El juicio de expertos consiste en seleccionar a 3 personas con la mayor experiencia posible en el campo que engloba al proyecto y que ellos estimen el tiempo que cada tarea necesitará. Tras esto se realiza una media ponderada de los valores en base al nivel de experiencia de cada "juez" obteniendo así el valor final.

Los tres expertos que han realizado la estimación de las tareas son:

- Fernando Barber: Docente e investigador titular de la Universidad de Valencia. Parte del departamento de informática. Imparte varias asignaturas dentro del campo de las TIC y además es autor de varios estudios de simulación cardiaca por lo que posee experiencia en CARP. Además ha dirigido muchos TFGs a lo largo de su carrera profesional, incluido este, que también ayudará a que sus estimaciones sean exactas.
- Rafael Sebastián: Docente e investigador titular de la Universidad de Valencia. Parte del departamento de informática, también ha realizado estudios modelando y simulando la electrofisiología cardiaca en la UV por lo que además posee experiencia en los programas y en el campo de este trabajo final de grado.
- Ignacio García: Docente e investigador de la Universidad de Valencia. Tiene un trasfondo académico en matemáticas y un doctorado en ciencias de la computación. Posee además un alto conocimiento en el campo de la simulación cardiaca puesto que forma parte del CoMMLab de la UV (laboratorio computacional de simulación multiescala, en español) al igual que Fernando y Rafael.

Como se puede ver todos los expertos consultados están muy cualificados y conocen en profundidad los temas a tratar en el proyecto por lo que todas sus estimaciones son igualmente válidas. Sin embargo, se va a otorgar una ponderación algo mayor a Fernando, principalmente ya que al ser el tutor de este TFG conoce más en detalle las características del proyecto por lo que sus valores serán más precisos

- Fernando Barber: α = 0.4
- Rafael Sebastián: α = 0.3
- Ignacio García: α = 0.3

Ya obtenidas las estimaciones de cada experto se pasa al cálculo final del tiempo necesario para cada tarea y por consiguiente de la totalidad del proyecto como se ha mencionado antes mediante una media ponderada cuya fórmula es:

$$T = T_{Fernando} * 0.4 + T_{Rafael} * 0.3 + T_{Ignacio} * 0.3$$

Los resultados son los siguientes:

Número	Nombre de la Tarea	Estimación F. Barber	Estimación I. García	Estimación R. Sebastián	Media Ponderada
1	**Propuesta del Proyecto**	**7**	**26**	**20**	**16.6**
1.1	Objetivos y especificaciones	4	8	8	6.4
1.2	Cronograma	2	16	4	6.8
1.3	Aprobación del proyecto	1	2	8	3.4
2	**Estado del Arte**	**68**	**56**	**100**	**74.0**
2.1	El corazón	20	12	24	18.8
2.2	Simulación Cardiaca	24	20	36	26.4
2.3	Estudio del *Framework*	24	24	40	28.8
3	**Búsqueda y selección de herramientas**	**12**	**8**	**8**	**9.6**
4	**Planificación y especificación**	**30**	**32**	**32**	**31.2**
4.1	Análisis de Requisitos	8	8	8	8.0
4.2	Planificación Temporal	8	8	8	8.0
4.3	Planificación de Costes	10	12	8	10.0
4.4	Viabilidad Legal	4	4	8	5.2
5	**Análisis y Diseño**	**48**	**32**	**56**	**45.6**
5.1	Casos de Uso	8	12	8	9.2
5.2	Diagrama de Clases	40	20	48	36.4
6	**Implementación**	**96**	**148**	**104**	**114.0**
6.1	Instalación del Framework	12	8	8	9.6
6.2	Formación en el Framework	24	40	16	26.4
6.3	Desarrollo de la aplicación	60	100	80	78.0
7	**Pruebas**	**28**	**40**	**32**	**32.8**
7.1	Pruebas de requisitos funcionales	16	20	16	17.2
7.2	Pruebas de requisitos no funcionales	12	20	16	15.6
8	**Conclusiones**	**16**	**42**	**40**	**31.0**
8.1	Posibles cambios/mejoras	10	40	24	23.2
8.2	Cierre del Proyecto	6	2	16	7.8
Total		**305**	**384**	**392**	**354.8**

Tabla 3. Estimación temporal, en horas, mediante juicio de expertos sobre las tareas del proyecto.

Con los resultados de la tabla 3 pasamos a obtener el diagrama de Gantt. Este diagrama permite visualizar de forma clara cuando cada tarea debe realizarse así como las relaciones de dependencia entre ellas. De esta forma podemos planificar el orden de realización de cada tarea de forma eficaz y priorizar las tareas críticas del proyecto, esto se conoce como el camino crítico donde si alguna de las tareas que lo conforman se retrasa el resto del proyecto también se retrasará. Sin embargo, debido a que este proyecto es realizado por una única persona ninguna tarea se puede realizar de forma concurrente por lo que todas las tareas forman parte del camino crítico.

En un diagrama de Gantt se suelen visualizar las tareas en días. En el diagrama se supone que una semana consta de 5 días laborables y cada día son 8 horas. En el caso de necesitar Por lo que las estimaciones temporales que necesitaremos para el diagrama son:

- Propuesta del proyecto: 3 días
- Estado del arte: 11 días
- Búsqueda y selección de herramientas: 2 días
- Planificación y especificación: 5 días
- Análisis y diseño: 7 días
- Implementación: 16 días
- Pruebas: 5 días
- Conclusiones: 4 días
- Total: 53 días

Tarea	Duración (Días)	F. Inicio	F. Fin	Semana 1	Semana 2	Semana 3
1. Propuesta del Proyecto	**3**	**07/05/2018**	**09/05/2018**			
1.1 Objetivos y especificaciones	1	07/05	07/05			
1.2 Cronograma	1	08/05	08/05			
1.3 Aprobación del proyecto	1	09/05	09/05			
2. Estado del Arte	**11**	**10/05/2018**	**24/05/2018**			
2.1 El corazón	3	10/05	14/05			
2.2 Simulación Cardiaca	4	15/05	18/05			
2.3 Estudio del *Framework*	4	21/05	24/05			
3. Búsqueda y selección de herramientas	**2**	**25/05/2018**	**28/05/2018**			
4. Planificación y especificación	**5**	**29/05/2018**	**04/06/2018**			
4.1 Análisis de Requisitos	1	29/05	29/05			
4.2 Planificación Temporal	1	30/05	30/05			
4.3 Planificación de Costes	2	31/05	01/06			
4.4 Viabilidad Legal	1	04/06	04/06			
5. Análisis y Diseño	**7**	**05/06/2018**	**13/06/2018**			
5.1 Casos de Uso	2	05/06/2018	06/06/2018			
5.2 Diagrama de Clases	5	07/06/2018	13/06/2018			
6. Implementación	**16**	**14/06/2018**	**05/07/2018**			
6.1 Instalación del Framework	2	05/06	06/06			
6.2 Formación en el Framework	4	07/06	12/06			
6.3 Desarrollo de la aplicación	10	13/06	26/06			
7. Pruebas	**5**	**24/09/2018**	**28/09/2018**			
7.1 Pruebas de requisitos funcionales	3	24/09	26/09			
.2 Pruebas de requisitos no funcionales	2	27/09	28/09			
8. Conclusiones	**4**	**01/10/2018**	**04/10/2018**			
8.1 Posibles cambios/mejoras	3	01/10	03/10			
8.2 Cierre del Proyecto	1	04/10	04/10			

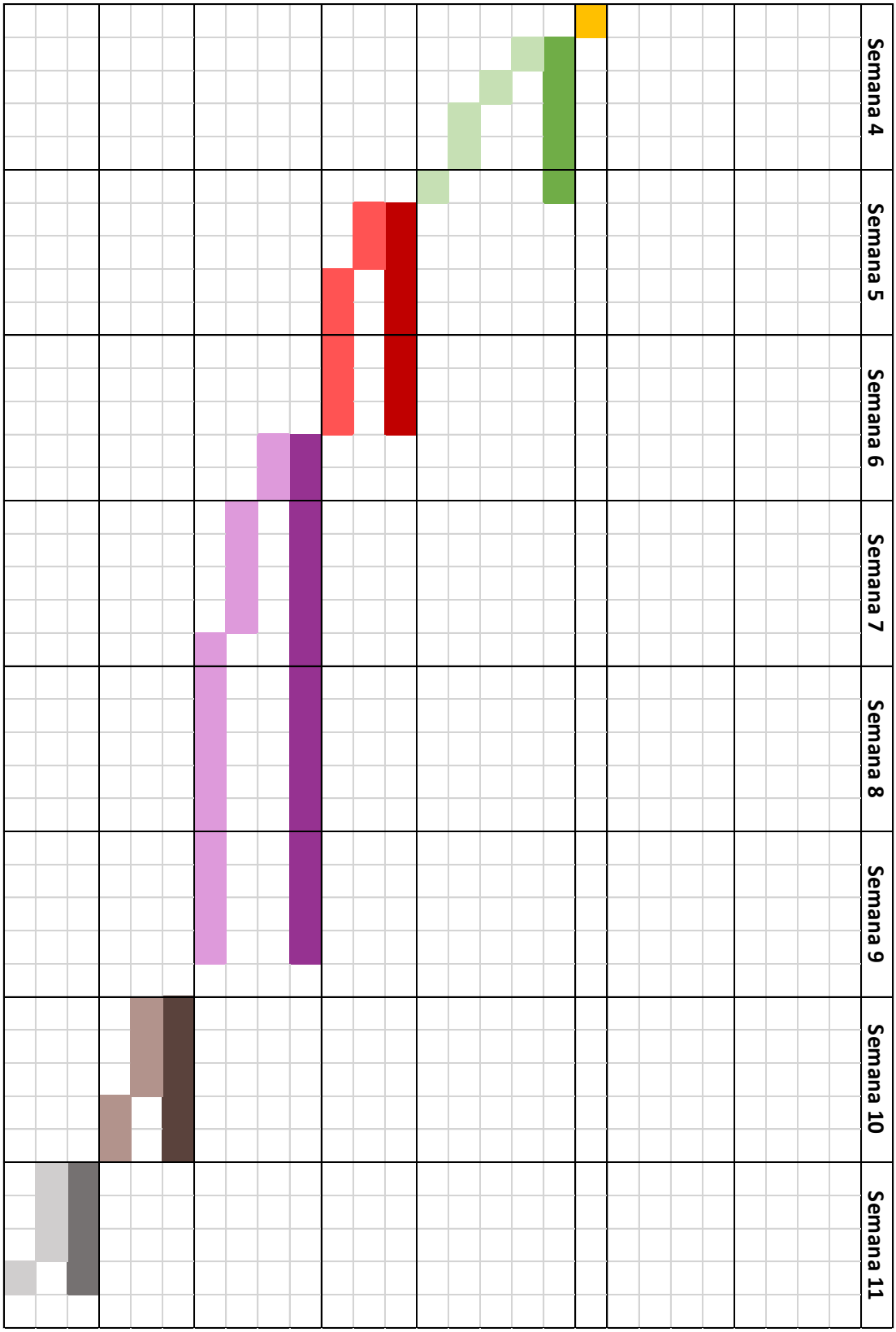

Tabla 4. Diagrama de Gantt del proyecto

5.3.3 Planificación de costes

Se tratara de obtener una estimación económica del coste del proyecto, teniendo en cuenta sueldos, uso de herramientas y/o costes ocultos entre otros.

Costes de mano de obra

En primer lugar vamos a calcular el salario que supuestamente deberíamos de percibir por la realización del proyecto. El salario hipotético se ha obtenido en base a la *Guía del Mercado Laboral 2018* [62] un estudio elaborado por Hays (empresa de selección de personal) donde se encuestaron a más de 1,000 empresas y 8,000 trabajadores.

Esta guía proporciona los salarios anuales brutos medios para varias profesiones del sector de las tecnologías de la información en varias ciudades de España y para diferentes años de experiencia. Para el cálculo de costes se usaran los sueldos en Valencia con una experiencia de 0 a 2 años. Además se usara la aproximación estándar de 1780 horas por año laboral y un 8% de contingencia.

Tarea	Posición	Salario/Hora	Horas	Salario Total	Coste Total (€)
Propuesta del Proyecto	Development Manager	22.47 €	16.6	373.00 €	402.84 €
Estado del Arte	Development Manager	22.47 €	74.0	1662.78 €	1795.80 €
Búsqueda y Selección de Herramientas	Development Manager	22.47 €	9.6	215.71 €	232.97 €
Planificación y especificación	Software Architect	20.79 €	31.2	648.65 €	700.54 €
Análisis y Diseño	Software Architect	20.79 €	45.6	948.02 €	1023.87 €
Implementación	Desarrollador C/C++	12.92 €	114.0	1472.88 €	1590.71 €
Pruebas	QA	12.92 €	32.8	423.78 €	457.68 €
Conclusiones	Development Manager	22.47 €	31.0	696.57 €	752.30 €

% de Contingencia
8%

Total	6,956.70 €

Tabla 5. Desglose de los gastos de mano de obra

Coste del material

Por costes materiales se entienden los gastos incurridos de la compra de licencias, hardware, software, etc., y teniendo en cuenta las horas que se van a gastar para el proyecto en relación con las horas totales de uso esperado. Además, volvemos a aplicar un 8% extra al coste como contingencia.

Activo	Coste	Tiempo de Uso (h)	Coste Amortizado
MSI GE60 20E-223XES	777.15 €	276.4	26.07 €
Módulo RAM Silicon Power 4GB	16.74 €	276.4	0.56 €
Ubuntu 18.04	- €	146.8	- €
NetBeans IDE 8.2	- €	146.8	- €
Visual Paradigm 14.1	- €	31.2	- €
Licencias	- €	-	- €

Total	26.63 €

Tiempo de Amortización	% de Contingencia
5 años	8%

Tabla 6. Desglose de los costes del material.

Se ha fijado como valor de amortización del portátil a usar en 5 años, el cual posee las siguientes características técnicas:

- Procesador: Intel Core i5-4200M @ 2.50GHz 2 núcleos (4 virtuales)
- Tarjeta Gráfica 1: Intel HD Graphics 4600
- Tarjeta Gráfica 2: NVIDIA GeForce GTX 765M (averiada)
- Memoria RAM: 2x4GB DDR3 SDRAM 1600MHz
- Disco Duro: 500GB
- Sistema Operativo: Windows 10 y Ubuntu

Coste de venta

Si bien se han calculado el precio que costaría realizar el proyecto falta tener en cuenta aspectos como los gastos indirectos, un 50% extra sobre el precio de mano de obra; el beneficio esperado del proyecto, en este caso del 20%, y el IVA del 21% para poder calcular el precio de venta al público, es decir, el precio final por el que habría que vender el software.

HOJA DE ESTIMACIÓN ECONÓMICA DE PROYECTO

DATOS DEL PROYECTO

TITULO:	Conversor de mallas 3D de fibras de Purkinje para el modelado del corazón en CARP	REFERENCIA:	1
CLIENTE:	Universidad de Valencia	RESPONSABLE:	Víctor Guillermo Andrés Escudero
FECHA INICIO:	07/05/2018	FECHA FIN:	09/07/2018

PRECIO VENTA (SIN IVA):	12,554.02 Euros
ESFUERZO:	354.8 Horas
MARGEN BENEFICIOS:	20.00 %

1. ESTIMACIÓN DE COSTES Y GASTOS

1.A PERSONAL

PORCENTAJE DE COSTES INDIRECTOS (OVERHEAD) 50%

PERSONAL	HORAS TR.	COSTE D./H	COSTE T./H	COSTE	% CONTING	COSTE FINAL
Manager de Desarrollo	131.2	22.47 €	33.71 €	4,422.10 €	8.00%	4,775.86 €
Arquitecto de Software	76.8	20.79 €	31.19 €	2,395.01 €	8.00%	2,586.61 €
Desarrollador C/C++	114	12.92 €	19.38 €	2,209.32 €	8.00%	2,386.07 €
QA	32.8	12.92 €	19.38 €	635.66 €	8.00%	686.52 €
SUBTOTAL	**354.8**	**18.15 €**	**27.23 €**	**9,662.09 €**	**8.00%**	**10,435.06 €**

1.B AMORTIZACION HW/SW

ELEMENTO HW / SW	PRECIO/UD	F. COMPRA	FIN AMOR	UNIDADES	COST Ud/H	HORAS USO	COSTE	% CONTING	COSTE FINAL
Portatil	777.12 €	01/01/2018	31/12/2022	1	0.09 €	276.4	24.13 €	8.00%	26.07 €
Memoria RAM	16.74 €	01/01/2018	31/12/2022	1	0.00 €	276.4	**0.52 €**	8.00%	0.56 €
			SUBTOTAL	**2**	**0.04 €**	**276**	**24.65 €**	**8.00%**	**26.63 €**
						TOTAL COSTES Y GASTOS	**9,686.74 €**	**7.41%**	**10,461.68 €**

2. PRECIO DE VENTA

2.A MARGEN

		SIN CONTINGENCIAS		CON CONTINGENCIAS	
CONCEPTO	MARGEN	IMPORTE	V. MARGEN	IMPORTE	V. MARGEN
PERSONAL	20.00%	9,662.09 €	1,932.42 €	10,435.06 €	2,087.01 €
AMORTIZACION HW/SW	20.00%	24.65 €	4.93 €	26.63 €	5.33 €
PORCENTAJE MARGEN PROYECTO	20.00%		**1,937.35 €**		**2,092.34 €**

2.B PRECIO DE VENTA

PRECIO DE VENTA, Euros, SIN ESCALACION INTERANUAL			**11,624.09 €**	**12,554.02 €**
PRECIO DE VENTA, Euros, CON IVA al ...	**21.00%**	**TOTAL**	**14,065.15 €**	**15,190.36 €**

Tabla 7. Coste del Proyecto

5.3.4 Viabilidad económica

El principal indicador de si un proyecto es viablemente económico es si se considera plausible su venta por el precio de venta al público estimado en la planificación de costes. Debido a la naturaleza del proyecto los únicos clientes potenciales son grupos de investigación cardiaca en CARP y que además hayan usado VTK para definir sus mallas de fibras de Purkinje. Por lo tanto la lista de compradores es muy limitada, además si se pretende vender el software habría que obtener una licencia comercial de CARP aumentando considerablemente el precio del proyecto.

Por todo eso podemos concluir que el proyecto no es económicamente viable, sin embargo, se trata de un trabajo realizado por un estudiante como parte de su formación por lo que no se desea obtener ningún beneficio económico, además ni es necesario que las horas de trabajo sean remuneradas ni adquirir una licencia comercial de CARP por todo esto se va a seguir adelante con el proyecto.

5.3.5 Viabilidad legal

El proyecto no presenta, a día de hoy, ningún problema legal, o normativa específica a la que se deba someter.

En el caso de que en un futuro se pueda digitalizar las fibras de Purkinje de un paciente humano esta información seguiría sin estar sometida al Reglamento General de Protección de Datos (RGPD) [63], la cual define en su artículo 4.1 los datos personales como “toda información sobre una persona física identificada o identificable”. Esto se debe a dos razones:

- La aplicación solo necesitará como datos de entradas una malla por lo que sería posible excluir los datos personales asociados a esa malla antes de ser usadas por el programa. De esta forma la información de la malla no pertenece a ninguna persona identificada.
- Aunque se podría argumentar que la disposición de las fibras de Purkinje es única a cada persona y que podría ser usada en un futuro para identificar la persona a la que pertenece, es decir, información perteneciente a una persona identificable; la consideración número 26 del RPGD dice que para saber si una persona es identificable “deben tenerse en cuenta todos los medios, […] que razonablemente pueda utilizar […] cualquier otra persona para identificar […] a la persona física. Para determinar si existe una probabilidad razonable de que se utilicen medios para identificar a una persona física, deben tenerse en cuenta todos los factores objetivos, como los costes y el tiempo necesarios para la identificación”. Esta última oración excluiría claramente esta información como datos personales.

6 Análisis y Diseño

En este apartado se van a analizar en profundidad los requisitos que tienen que cumplir la aplicación, además de cómo estos afectan al diseño de la aplicación, que a su vez condiciona de forma directa como se implementará la aplicación.

6.1 Casos de Uso

Primero se van a describir y analizar los casos de uso. Estos especifican las acciones que la aplicación puede hacer y cuyo resultado se puede observar y tienen valor para un actor (usuarios de la aplicación). Los casos de uso de este programa se pueden observar en el siguiente diagrama de casos de uso y en sus plantillas extendidas.

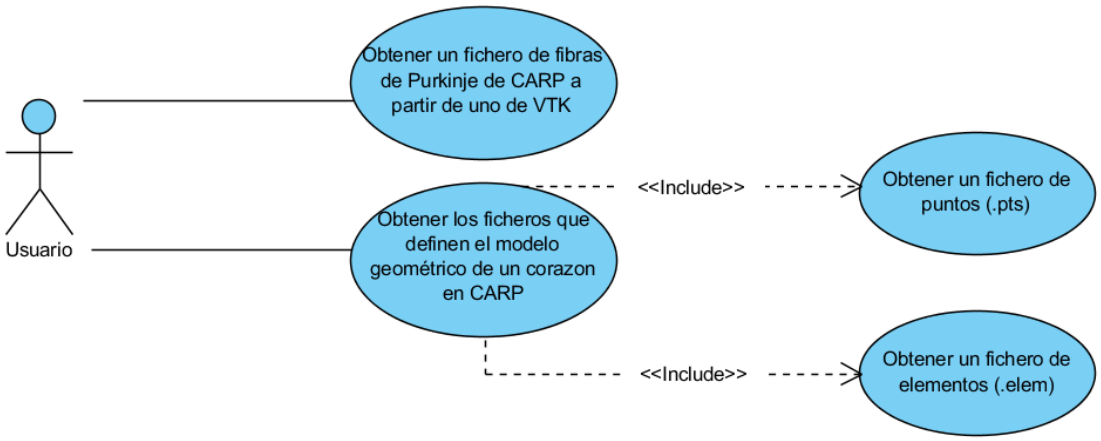

Ilustración 17. Casos de uso de la aplicación.

6.1.1 Plantillas de Casos de Uso Extendidas

CU01	Obtener un fichero de fibras de Purkinje de CARP a partir de uno de VTK.
Versión	1
Autor	Víctor Andrés
Actor Principal	Usuario
Descripción	El sistema lee y parsea el fichero VTK introducido por el usuario, después crea las estructuras de datos necesarias y tras calcular toda la información restante necesaria, crea el fichero de salida y escribe en el los datos en el formato adecuado.
Precondiciones	El fichero de entrada es un fichero correcto y adecuadamente formateado.
Postcondiciones	Fichero con extensión .pkje correctamente formateado de acuerdo a las especificaciones de CARP
Comentarios	

Tabla 8. Plantilla extendida del C.U que genera un fichero de fibras de Purkinje

CU02	Obtener los ficheros que definen el modelo geométrico del corazón en CARP
Versión	1
Autor	Víctor Andrés
Actor Principal	Usuario
Descripción	El sistema lee y parsea el fichero VTK introducido por el usuario, después crea las estructuras de datos necesarias, crea los dos ficheros de salida necesarios y los escribe al disco.
Precondiciones	El fichero de entrada es un fichero correcto y adecuadamente formateado.
Postcondiciones	Dos ficheros compatibles con CARP uno de puntos (.pts) y otro de elementos (.elem) con el mismo nombre.
Comentarios	

Tabla 9. Plantilla extendida del C.U que genera los ficheros de un modelo geométrico del corazón en CARP.

CU03	Obtener el fichero de puntos (.pts) de CARP.
Versión	1
Autor	Víctor Andrés
Actor Principal	Usuario
Descripción	El sistema lee y parsea el fichero VTK introducido por el usuario, después crea el fichero de salida y escribe en él las coordenadas cada uno de los puntos.
Precondiciones	El fichero de entrada es un fichero correcto y adecuadamente formateado.
Postcondiciones	Un fichero de puntos formateado de acuerdo a la especificación de CARP.
Comentarios	

Tabla 10. Plantilla extendida del C.U que genera el fichero de puntos en CARP.

CU04	Obtener el fichero de elementos (.elem) de CARP.
Versión	1
Autor	Víctor Andrés
Actor Principal	Usuario
Descripción	El sistema lee y parsea el fichero VTK introducido por el usuario, después crea el fichero de salida y escribe en él las coordenadas cada uno de los puntos.
Precondiciones	El fichero de entrada es un fichero correcto y adecuadamente formateado.
Postcondiciones	Un fichero de puntos formateado de acuerdo a la especificación de CARP.
Comentarios	

Tabla 11. Plantilla extendida del C.U que genera el fichero de elementos en CARP.

6.2 Diagrama de Clases

En este apartado se mostrara el diagrama de clases que sirve para modelar las acciones que el sistema realiza. En él se pueden observar los conceptos y elementos más importantes y que tareas pueden hacer.

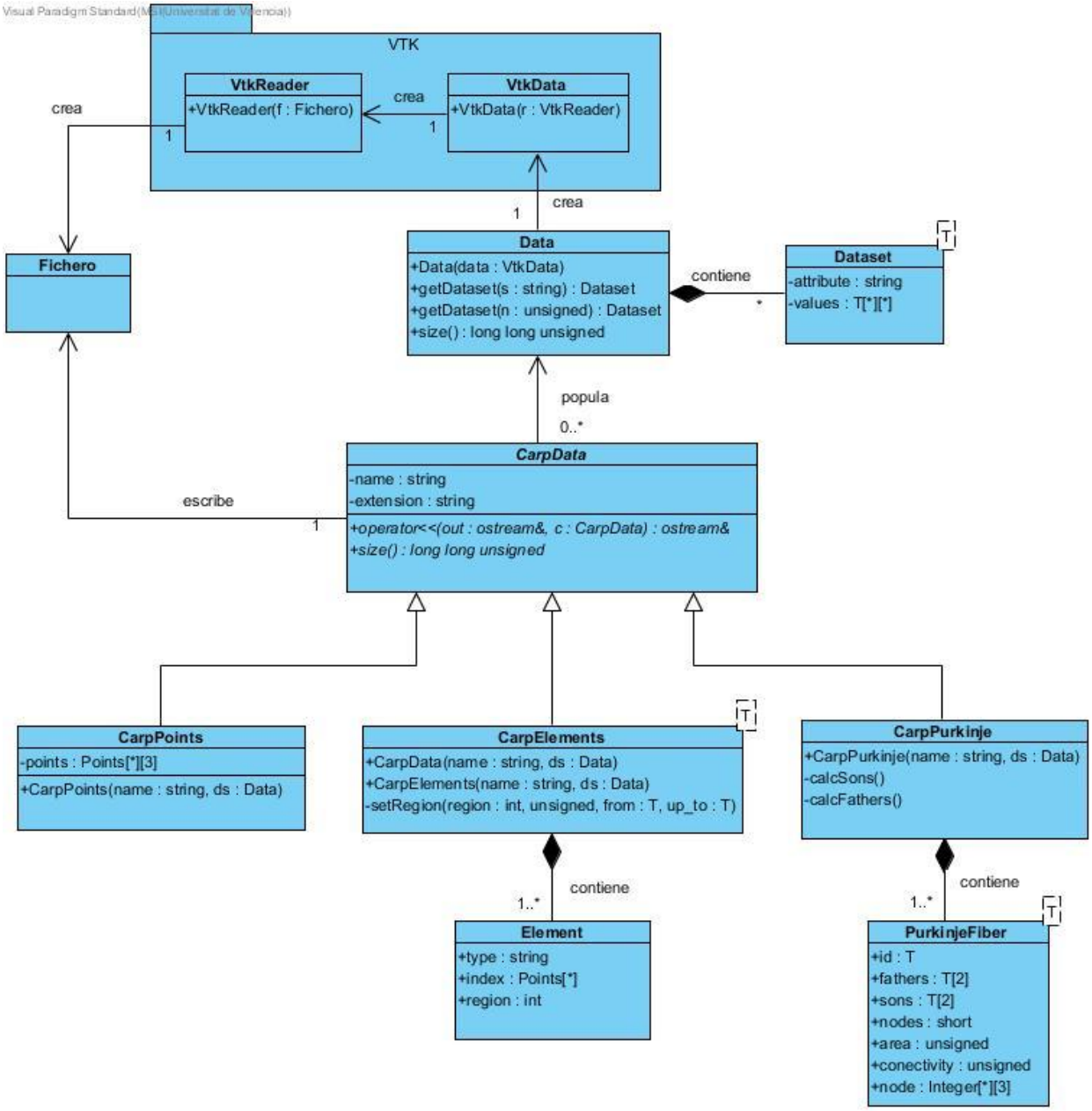

Ilustración 18. Diagrama de clases de la aplicación.

6.2.1 Clases de la aplicación

A continuación se verá una pequeña explicación de la función y los objetivos de cada una de las clases de la aplicación que se pueden ver en el diagrama de clases de más arriba.

- **Fichero:** Representan un fichero en el ordenador. La aplicación lee un fichero de entrada de VTK y como da como salida 1 o más ficheros listos para ser usados en CARP.
- **VtkReader** y **VtkData:** Son las clases ya implementadas por VTK. La primera parsea un fichero en formato VTK y genera una instancia de VtkData con la toda la información geométrica del archivo.
- **Data** y **Dataset:** Clase propia que contiene toda la información disponible en VtkData de forma que sea fácil de acceder.
- **CarpData:** Clase abstracta que sirve como clase base para cada uno de los ficheros que la aplicación soporta. Posee los métodos abstractos puros de size() y sobrecarga del operador <<, por lo que las clases que deriven de esta tendrán que tener su propia implementación.
- **CarpPoints**, **CarpElements** y **CarpPurkinje:** Representan cada uno de los tipos de ficheros que la aplicación puede convertir a partir de un fichero en VTK.
- **Element** y **PurkinjeFiber:** Estructuras auxiliares, almacenan toda la información sobre el elemento que la clase que las contienen representa.

6.2.2 Justificación del diseño

Aquí se explicaran las decisiones que se han tomado mientras se realizaba el diseño de la aplicación así como que objetivo se pretende conseguir con estas medidas y como se han implementado en el diseño. Hay 3 aspectos principales:

Primero, tanto VTK como CARP son programas en constante desarrollo por lo que no hay ninguna garantía de que el programa sea siempre válido. Por lo tanto si alguna versión nueva modifica como VTK almacena sus conjuntos de datos, o se cambia el formato que usa CARP para sus archivos, por poner algunos ejemplos, habrá que modificar el programa.

Para que estos cambios se puedan implementar de forma fácil y modificando el menor código posible se ha creado la clase Data. Esta clase actúa a modo de adaptador entre VTK y CARP. Si no existiera esta clase y ocurriera alguna modificación en la clase de VtkData habría que modificar los constructores de todos y cada uno de los ficheros soportados por el programa. De esta forma garantizamos que la estructura de Data sea estable en el tiempo y solo habría que modificar un método (su constructor).

Segundo, está la razón por la que se usa la clase abstracta CarpData, principalmente es para que el programa sea mucho más flexible y se mantenga encapsulado con el uso de polimorfismo. Gracias a esto si en un futuro se desea que el programa soporte la conversión a algún tipo de fichero adicional, por ejemplo un fichero de definición de pulsos, que simplemente baste con añadir una clase que extienda de CarpData, sin necesidad de modificar el resto del código.

Tercero, la parametrización de ciertas estructuras, un problema potencial que se desea evitar con este enfoque es que la memoria usada por el programa sea excesiva. Debido a la naturaleza de los datos que trata la aplicación puede haber desde solo algunos cientos de datos hasta varios millones de elementos. Con el uso de plantillas garantizamos que se use el tipo de dato (int, short, long) menor necesario, y aunque por ejemplo solo ahorremos 2 bytes de usar un short a un int, cuando tenemos en cuenta que esto se produce para cada todos y cada uno de los elementos se vuelve un ahorro sustancial.

7 Implementación

En esta sección se va a expandir sobre el proceso de implementación de la aplicación. Se van a discutir y explicar cómo se ha decidido hacer las distintas partes del programa, las decisiones que se han tomado en cada una de las partes, así como que problemas han surgido y como se han solucionado.

7.1 Datasets

Estas clases son las encargadas de actuar como puente entre los archivos vtk y los archivos de CARP. Estas clases almacenan los datos extraídos del fichero vtk de forma que sean fáciles de trabajar con ellos, también proporcionan acceso a los datos almacenados.

7.1.1 Patrón de diseño: *Multiton*

Este conjunto de clases son de vital importancia para el correcto funcionamiento de la aplicación, ya que básicamente todas las clases las usan. O bien, para almacenar los datos extraídos del fichero vtk, o bien, para obtener estos datos y trabajar sobre ellos.

Para facilitar el uso por el resto de clases se decidió implementar estas clases siguiendo un patrón de diseño muy similar al conocido como *Singleton*. Básicamente se trata de en este caso de una *hash table* estática donde se van almacenando las diferentes instancias que se crean a lo largo de todo el programa. Tras esto se crea una función también estática a la que se le proporciona una llave y esta devuelve un puntero a la instancia si existe.

Este patrón tiene la desventaja importante de que es conceptualmente similar al uso de variables globales, una práctica muy desaconsejada. Sin embargo, como se ha explicado antes esta clase es usada por prácticamente todo el programa, por lo que este patrón tiene sentido. En caso contrario habría que pasar los conjuntos de datos (*datasets*) como parámetros de la mayoría de funciones, con este enfoque simplificamos el código necesitando solo llamar a la función estática que devuelve un puntero con acceso al conj. de datos deseado.

7.1.2 Parametrización de la clase Dataset

Como se ha visto antes, un aspecto clave de este proyecto era el de almacenar internamente los datos suministrados por el archivo con el tipo de dato de menor tamaño sin perder precisión. Para conseguir este objetivo se parametrizo la clase Dataset de tal forma que se pueda adaptar a cualquier tipo de dato con el uso de plantillas.

7.1.3 Polimorfismo

Gracias al uso de plantillas se ha conseguido lo que se conoce como polimorfismo estático (en tiempo de compilación), sin embargo esto plantea un problema, ya que no es hasta en tiempo de ejecución cuando sabemos el tipo de datos a usar ya que esta información se

encuentra en los ficheros de entrada. La forma más directa para solventar este problema es el de incluir una larga lista de if else similar a este pseudocódigo:

```
if (data_type == float)
        Dataset<float> d;
        HacerAlgo(d);
else if (data_type == int)
        Dataset<int> d;
        HacerAlgo(d);
[…]
```

El problema es que el ámbito de estas variables está limitado a las clausulas if/if else por lo que no es posible acceder a ellas fuera de su instanciación, haciéndolas prácticamente inútiles, o se necesita copiar las funciones para cada uno de los posibles tipos de datos. Esta solución es obviamente una mala idea ya que dificulta enormemente la lectura y mantenimiento del código y la filosofía del polimorfismo.

La solución que se implementó finalmente fue mediante el uso de herencia y funciones virtuales para conseguir también polimorfismo dinámico. Se hizo que la clase Dataset heredara de la clase DatasetAbstract. De esta forma se pueden almacenar cualquier tipo de Dataset con un puntero de la clase DatasetAbstract, y ya en tiempo de ejecución se invocará la función correspondiente al tipo de datos del conj. de datos con el que está trabajando.

7.1.4 Patrón de diseño: *Factory method*

El problema del polimorfismo esta ya por lo tanto prácticamente arreglado, sin embargo, falta un detalle, c++ es un lenguaje fuertemente tipado por lo que es necesario saber el tipo de dato con el que se va a trabajar en tiempo de compilación esto implica que aunque ya no haría falta repetir el uso de las mismas funciones para cada uno de los tipos, sí que hace falta algo así:

```
DatasetAbstract* d;
if (data_type == float)
        d = new Dataset<float>();
else if (data_type == int)
        d = new Dataset<int>();
[…]
HacerAlgo(d);
```

En un primer lugar esta tarea era parte del constructor de Dataset, sin embargo, hay un patrón conocido como método factoría (*Factory method)* que delega la responsabilidad de qué tipo de objeto se debe de crear a un método estático. De esta forma se consigue mantener el constructor de la clase más sencillo y separar la lógica de la elección del tipo de objeto a instanciar del constructor al método mencionado.

7.1.5 Estructura de datos seleccionada

Para almacenar internamente los datos extraídos del fichero VTK se ha optado por un vector de vectores, de la librería estándar de C++11. Esto se debe a que de los 3 tipos de ficheros de CARP actualmente soportados todos almacenan la información de forma secuencial. No se podía usar arrays ya que su tamaño solo puede saberse en tiempo de ejecución.

7.1.6 Uso de punteros

Como se puede ver el uso de polimorfismo dinámico se basa en usar un puntero de la clase padre que apunte a una instancia concreta de una de sus clases hijo. Sin embargo, el uso de punteros conlleva ciertos riesgos, ya que hay que asegurarse de evitar referencias colgantes (*dangling reference*) y/o fugas de memoria (*memory leaks*). Para evitar estos problemas se trató de usar al principio punteros inteligentes, sin embargo, daban problemas al intentar inicializarlos en el propio constructor de forma que apuntaran a *this* (necesario para almacenar el puntero en la tabla hash). Pero, como debido a la naturaleza del programa estos riesgos no están presentes se usaron finalmente punteros estándar. Por un lado los punteros al estar almacenados en una hash table estática su ámbito es global por lo que nunca serán eliminados, esto imposibilita la ocurrencia de fugas de memoria. También como los conjuntos de datos a los que apuntan estos punteros son usados a lo largo de todo el ciclo de vida del programa, no existe la necesidad de tener que eliminar el conjunto de datos para liberar memoria por lo que tampoco habrán referencias colgantes.

7.2 VtkParser

Esta clase es relativamente sencilla, simplemente se usan las clases proporcionadas por VTK para leer el fichero y generar un Dataset donde se almacena toda la información del fichero. Tras esto se obtiene información como las coordenadas de los puntos, los elementos e información adicional y se almacenan en instancias de Dataset.

Sin embargo, aun siendo sencilla el principal problema para la implementación de esta clase fue el de familiarizarse con la librería VTK. Esta, al ser una librería tan potente y capaz de realizar tareas tan complicadas como renderizado de escenas 3D, implica que existen cientos de funciones disponibles, cadenas de herencia entre clases de varios niveles, clases auxiliares, etc. Esto hizo que buscar la información necesaria en la documentación fuera una tarea más complicada de lo esperado.

Por último, un objetivo importante que se deseaba conseguir era el de almacenar de forma eficiente la información extraída del fichero de entrada. Mientras que VTK proporciona funciones que proporcionan el tipo de dato interno para cada conj. de datos esta información se da mediante un entero, mientras que para crear una instancia de la clase Dataset propia se necesita un string con el tipo de dato a almacenar. Para lograr esto se creó la función vtkDataTypeToNative que devuelve el string adecuado de acuerdo al fichero vtkType.h que define que primitiva representa cada entero.

7.3 Outputs

Estas clases son representaciones de cada uno de los distintos ficheros que la aplicación puede generar. En primer lugar está la clase AbstractFile que representa un fichero genérico y por lo tanto es bastante sencilla. Posee dos atributos que representan el nombre y la extensión del fichero, así como funciones muy sencillas como *getters* y *setters*. El objetivo de esta clase es principalmente lograr, una vez más, polimorfismo para facilitar la adición de nuevos ficheros.

7.3.1 Clases concretas

El comportamiento general de estas clases es el de primero obtener de los Dataset obtener los datos necesarios para poder crear el fichero. En el caso de que la información disponible proveniente de los conjuntos de datos no sea suficiente, también se encargan de generar los datos adicionales ellos mismos.

7.3.2 CarpPurkinje

Esta es la clase encarga de generar los ficheros de fibras de Purkinje, así como el objetivo principal del proyecto y la clase más compleja de las 3 existentes.

Fichero de configuración

De acuerdo al simulador CARP, una fibra de Purkinje se define de acuerdo a varios atributos tal como se puede ver en el anexo 2. Concretamente hay 3 atributos (tamaño del cable, resistencia y conductividad) que ya que no pueden deducirse a raíz de la información geométrica no pueden obtenerse por los datos suministrados por el fichero VTK.

En un principio las variables que representaban estos atributos tenían asignados unos valores por defecto desde la propia clase, y si luego se deseaba modificar estos valores bastaba con modificarlos en el código. Sin embargo, esto implica que cada vez que se desee usar otros valores hay que volver a compilar el programa, una tarea que se ha hecho innecesaria con el uso de un fichero de configuración externo (config.cfg) que el programa lee para obtener estos valores en tiempo de ejecución.

La forma que el programa usa este fichero es: primero se comprueba si el fichero existe, en caso contrario, se crea el fichero con los valores por defecto para después leerlo y reasignar a las variables los valores del fichero. El hecho que se reasignen estos valores y no se inicialicen es importante ya que de esta forma si el fichero no puede ser creado, por ejemplo el usuario no tienen los permisos necesario, se usaran los valores por defecto de las variables.

Algoritmo que genera las relaciones entre fibras de Purkinje

Otros atributos importantes que debían de ser obtenidos son calcular los padres e hijos de cada fibra de Purkinje.

Primero hay que definir que se considera un padre e hijo de una fibra de Purkinje. Una fibra está compuesta por una línea que va del punto A al B. Bien, pues cualquier otra fibra que

se origine en el punto B será hija de la fibra original, a la vez que el padre de esta fibra será la fibra original. De forma inversa ocurre para las fibras que finalicen el en punto A.

Uno de cosas que hay que tener en cuenta es que el orden en que aparecen las fibras de Purkinje no tiene por qué seguir ningún orden en concreto, por lo que en un principio para cada fibra que se quiera obtener sus relaciones se debe de iterar sobre todas las otras fibras. Este método por la fuerza bruta tienen un coste de ejecución de $O(n^2)$ y debido a la posibilidad de que hayan millones de fibras de Purkinje no es una opción muy atractiva.

La implementación final fue por lo tanto otra distinta. Suponiendo una fibra de Purkinje que va del punto A al B lo que hace el programa, concretamente para cada fibra se describa a continuación. Primero, busca si encuentra alguna fibra de Purkinje ya parseada que finalice en A, en este caso se ha identificado un padre de la fibra con la que se trabaja y un hijo en la fibra que se ha encontrado. Segundo, busca si hay alguna fibra que se origina en B, en este caso se ha encontrado un hijo de la fibra original y un padre de la fibra encontrada. Tercero, almacena el punto A junto con el índice de la fibra en un *unordered_multimap* (un tipo de tabla hash) de orígenes y el punto B y el índice en otra tabla hash de destinos.

Este último paso es clave, estas tablas hash están implementadas de tal forma que los puntos de origen o destino actúen como llave y el índice de la fibra de Purkinje al que corresponde ese origen/destino como valor. De esta forma evitamos tener que iterar sobre todas las fibras de Purkinje intentando buscar aquellas que se originen/finalicen en el punto deseado. Basta con buscarla en la tabla hash algo muy favorable ya que la complejidad media de búsqueda es constante O(1) así como la inserción. De esta forma se puede obtener de forma directa los índices de los cables que son hijos y/o padres. Todo esto hace que el algoritmo final sea mucho más eficiente con una complejidad lineal O(n), a costa de algo menos de eficiencia espacial.

Algoritmo corrector para fibras de Purkinje con más de 2 hijos

A lo largo del desarrollo del programa surgió un problema: algunas fibras tenían más de 2 hijos. Sin embargo, CARP no admite fibras con 3 o más hijos.

El algoritmo que se implementó para solucionar este problema fue el siguiente. Primero el programa itera sobre todas las fibras de Purkinje y almacena sus relaciones de vecindad de acuerdo al apartado anterior. Después, cuando se detecta alguna fibra con más de dos hijos la fibra padre se divide en dos cables, el cable original es ahora el 90% del cable original y el segundo el 10% final. Tras esto se asignan los dos primeros hijos originales como hijos del nuevo cable mientras que el tercer hijo y el cable nuevo se asignan como hijos del nuevo cable original. A continuación se puede ver una representación gráfica del algoritmo.

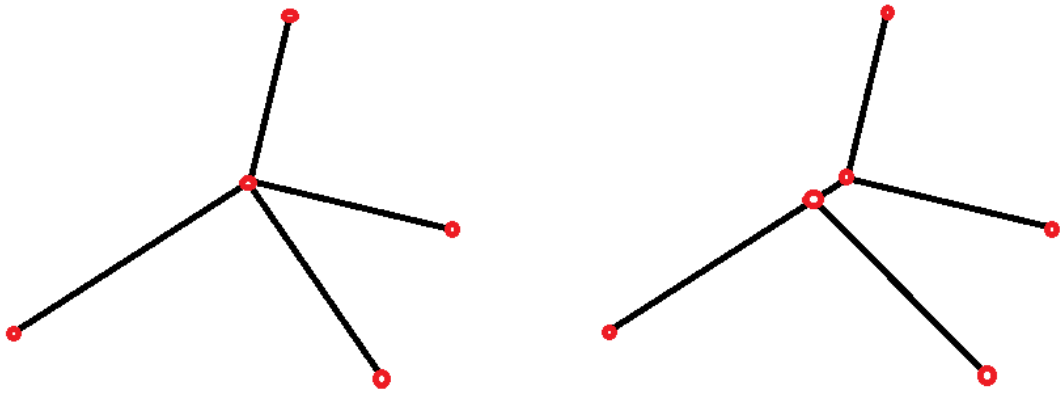

Ilustración 19. Representación gráfica del algoritmo que modifica las fibras de Purkinje con más de dos hijos. Izq. antes. Der. después.

7.3.3 Uso de punteros

Las 3 clases que generan los ficheros almacenan la información que obtienen de los Datasets mediante punteros de AbstractDataset. De esta forma logramos polimorfismo además de ahorrar copias redundantes de los conj. de datos, algo importante debido a que pueden llegar a ser objetos que ocupan mucha memoria.

7.3.4 Sobrecarga del operador inserción (<<)

A la hora de sobrecargar el operador de inserción ocurrió un problema, este se debe a que la función de sobrecarga es una función amiga y libre, por lo que no forma parte de la clase. Por lo tanto no se puede declarar como una función virtual impidiendo el polimorfismo de la clase.

Primero se intentó hacer que la función formara parte de la clase para poderla hacerla virtual, algo similar a:

virtual ostream& operator<< (ostream& out) const;

Sin embargo, esto implica que el operador << se tenía que usar en el orden inverso (output << cout) algo nada intuitivo. Para arreglar este problema se volvió a hacer amiga la función de tal forma que esta llamara a la función print() de la propia clase. Tras esto basta con declarar la función print() como virtual pura en la clase padre e implementar la función en las clases que heredan de esta, obteniendo así el resultado esperado.

7.4 Documentación

Como se ha explicado en el análisis de requisitos uno de ellos era el de crear la documentación del programa mediante el programa Doxigen, este manual de usuario puede encontrarse con el nombre de Documentacion.pdf junto con el programa.

Doxigen permite muchísimas opciones de configuración a la hora de generar la documentación, para simplificar el progreso se ha usado un *plugin* que permite generar el fichero de configuración con la ayuda de una interfaz gráfica, el fichero resultante también puede ser encontrado junto al programa con el nombre de DoxigenConfig.cfg. Las principales opciones que se han usado que no estuvieran activadas por defecto son:

OUTPUT_LANGUAGE = Spanish

EXTRACT_PRIVATE = YES

EXTRACT_STATIC = YES

GENERATE_RTF = YES

7.5 Diagrama de clases final

Con todos los cambios mencionados más arriba el diagrama de clases final es el siguiente:

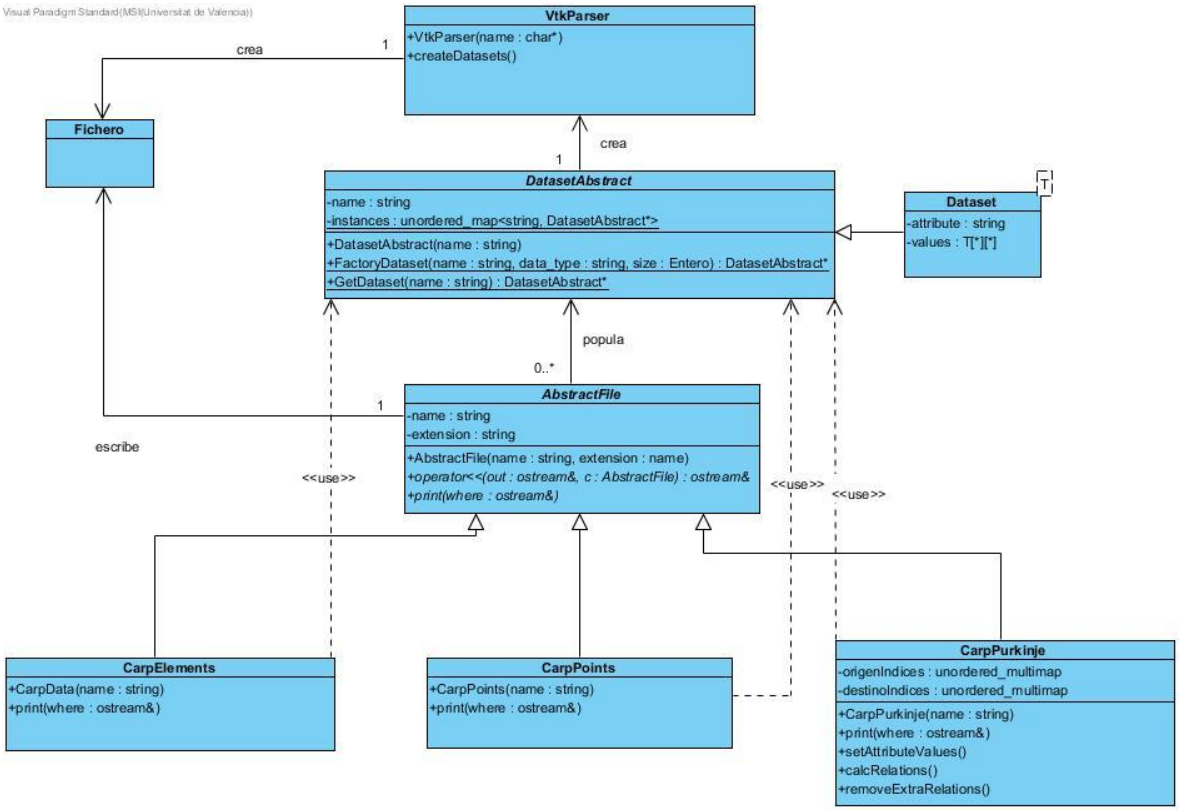

Ilustración 20. Diagrama de clases final.

8 Pruebas

Una vez completada la aplicación se ha realizado una serie de pruebas en el simulador cardiaco CARP para comprobar que los ficheros que genera el programa son válidos y se pueden usar satisfactoriamente.

Los ficheros con los que se trabajó fueron 1 modelo 3D de un corazón completo (aurículas y ventrículos) así como 3 modelos de fibras de Purkinje distintas compatibles con ese modelo cardiaco en particular. Después de generar los ficheros compatibles con CARP a partir de los ficheros de entrada ya mencionados se pasó a realizar las simulaciones sobre estos.

El simulador CARP como ya se ha visto necesita de ciertos ficheros específicos para realizar simulaciones, entre los que se encuentra el fichero de parámetros. Este fichero se encarga de dar valores al simulador todas las variables del experimento, como el tiempo total de la simulación, número de regiones, ruta donde se escriben los resultados, etc. Debido a la complejidad del simulador hay un número muy elevado de parámetros que se pueden modificar, aunque, por suerte, la mayoría ya tienen valores por defecto acordes a los que sería de esperar en una simulación estándar para un corazón sano. Por todo esto solo fue necesario modificar algunos parámetros concretos. La plantilla que se ha usado de base para estos experimentos se puede ver en el anexo 3.

Al principio se intentó simular un latido cardiaco en cada una de las 3 de fibras de Purkinje sobre el modelo del corazón. El simulador consiguió realizar las simulaciones, aunque no se obtuvieron los resultados esperados. Ningún cardiomiocito se excitaba por la corriente suministrada y el corazón se mantenía en reposo durante toda la simulación. Esto se debe a que la malla original y la generada por el programa estaban definidas con triángulos, y como se puede ver en la ilustración 15 este tipo de elementos solo están soportados parcialmente por CARP. El simulador considera esta malla como una única superficie 2D, en lugar de una malla volumétrica 3D, debido a esto, las simulaciones fallaban.

Sin embargo, sí que se pudo comprobar que los ficheros generados por el programa sí que se habían convertido correctamente. Cuando CARP realiza este tipo de simulaciones genera automáticamente una malla que combina la malla del corazón con la de las fibras de Purkinje. Por lo tanto se visualizó esta malla con la ayuda de mesher una herramienta de CARP para visualizar mallas. Los resultados se muestran a continuación.

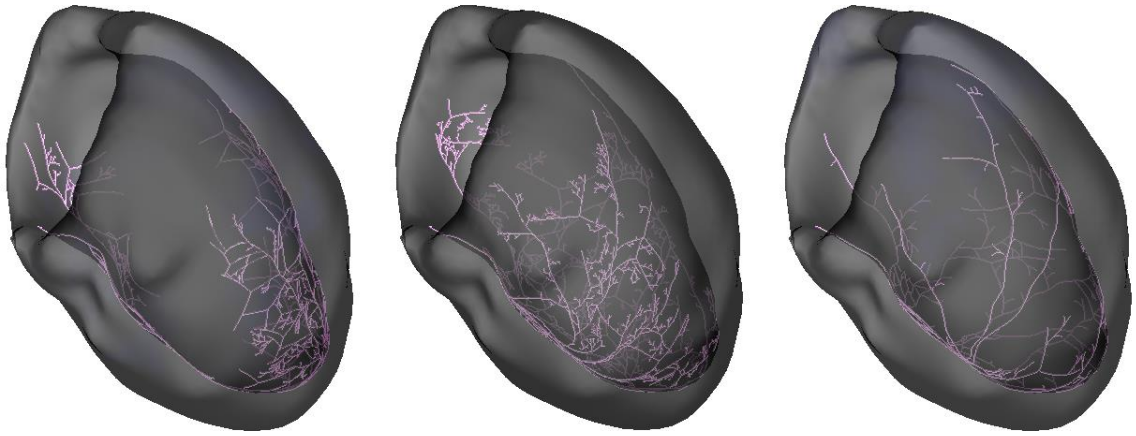

Ilustración 21. Visualización en meshalyzer de la malla 3D con 3 distribuciones de fibras de Purkinje distintas generadas por el programa a partir de ficheros en formato VTK.

Finalmente para confirmar la hipótesis de que las simulaciones no funcionaban debido a que la malla está compuesta por un tipo de elemento no soportado completamente por CARP se realizó una simulación alternativa donde la malla del tejido cardiaco estuviera formada por tetraedros que sí están planamente soportados.

Para asegurarse de que no hubiera ningún fallo o error en la malla, esta se creó mediante mesher, otra herramienta de CARP pensada para generar mallas simples automáticamente. Con este programa se creó una malla rectangular de 0.5x1x0.5 cm formada por tetraedros, formado por células de 300 μm^3. Esto último es importante para que la corriente eléctrica se pueda propagar correctamente entre células. Para esta malla se creó manualmente un fichero de fibras de Purkinje para realizar la simulación electrofisiológica. Los resultados esta vez sí que fueron los esperados por lo que se pudo confirmar que el conversor sí que genera los ficheros correctamente, solo que la malla compuesta por triángulos no es compatible con CARP. Los resultados de esta simulación se muestran más abajo.

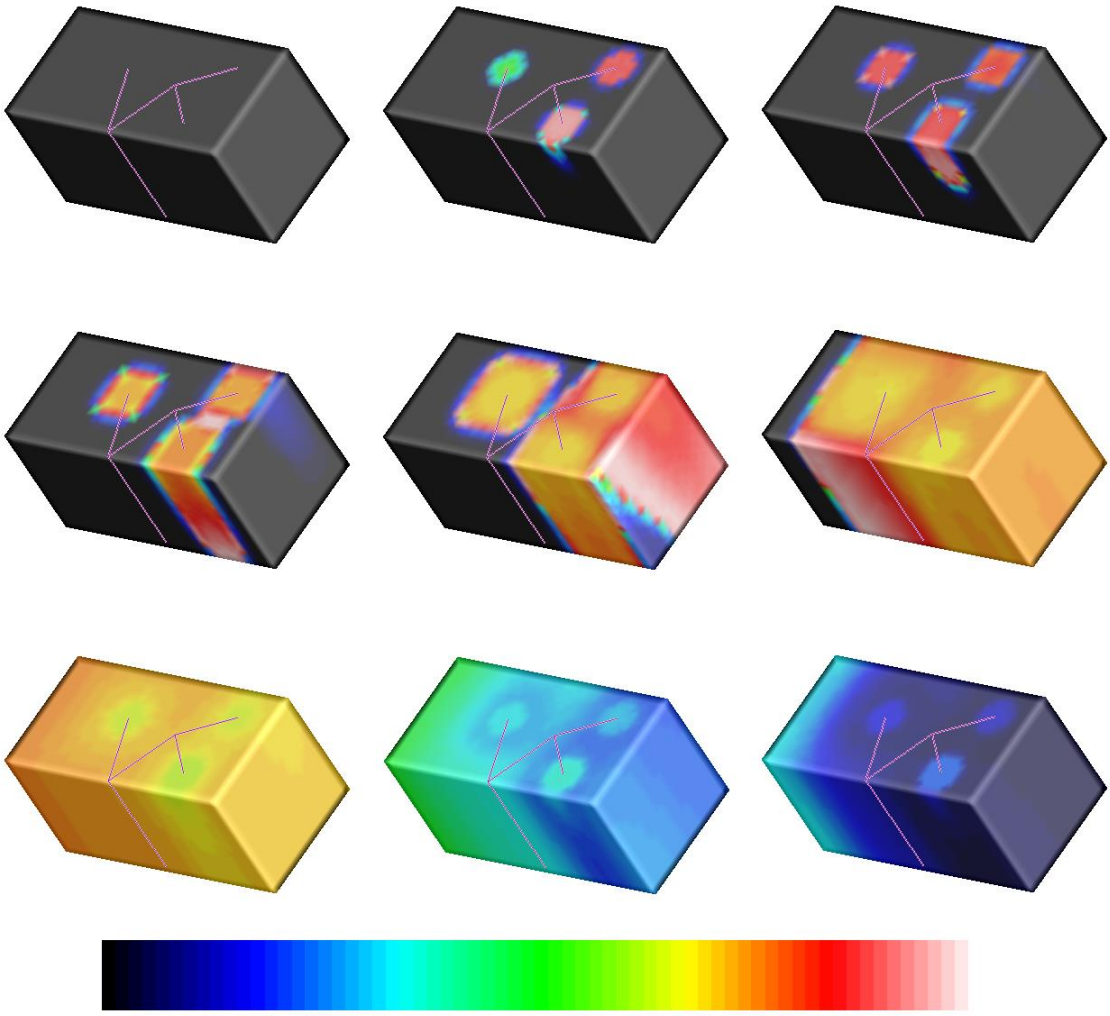

Ilustración 22. Arriba: Visualización de los voltajes de superficie debido a la excitación del haz de His para distintos instantes en el tiempo. Abajo: Escala de valores de voltajes, desde negro (-90mV) hasta blanco (+50mV).

9 Conclusiones y Trabajo Futuro

9.1 Conclusiones

En este proyecto se ha realizado una aplicación para Linux para el laboratorio de simulación computacional multiescala de la Universidad de Valencia. Tal que permita convertir ficheros en el formato nativo de VTK a ficheros de puntos, elementos o de fibras de Purkinje compatibles con el simulador CARP. Estos dos primeros aunque no estaban previstos al inicio del proyecto se han añadido para añadir más funcionalidades al programa.

Considero que el proyecto ha sido un éxito, tanto por que se han conseguido realizar simulaciones cardiacas satisfactorias con los ficheros generados por el programa, como por la gran cantidad de conocimientos adquiridos a lo largo del desarrollo del mismo. Lo más importante es que se ha obtenido una vista completa de lo que conlleva realizar una aplicación de esta escala. Y de como todos los conocimientos adquiridos a lo largo de la carrera han tenido que ser usados a lo largo del proyecto con mayor o menor importancia.

Destacar también el enfoque global que me ha aportado este proyecto, mientras que quizás los conocimientos adquiridos en clases se interiorizan de forma compartimentada, con la realización del proyecto se aprende como todos estos conocimientos se complementan unos a otros.

Finalmente, mencionar la gran cantidad de conocimientos que se han adquirido sobre muchas cosas que no tenía ni idea antes de realizar el proyecto, que sin duda serán de gran importancia en un futuro. Por ejemplo, me he familiarizado con VTK una de las librerías más completas e importantes que existen dentro del mundo del renderizado 3D y he aprendido cientos de cosas sobre el corazón y el increíblemente complejo mundo de la simulación cardiaca.

9.2 Trabajo Futuro

A lo largo de la realización del proyecto se han ido haciendo patentes posibles mejoras y/o cambios en el programa que beneficiarían al mismo. Que, sin embargo, no han podido ser implementadas debido a que el tiempo disponible no era ilimitado.

Una posible mejora sería la de eliminar algunas dependencias del programa, específicamente VTK. Se podría leer los ficheros de VTK de forma nativa, y almacenarlos en las estructuras de datos del programa directamente, sin necesidad de usar clases o funciones de la librería gráfica. Considero que esto es un cambio beneficioso por varios motivos. Primero, evita la necesidad de saber cómo funciona VTK, una librería bastante compleja, solo haría falta saber la sintaxis de los ficheros .vtk. Segundo, al no depender de VTK se puede usar en su lugar código propio que fuera más eficiente y se ajustara más a las necesidades del programa.

Otra mejora factible es la de implementar una interfaz gráfica para el programa. Mientras que la interfaz por línea de comandos actual es completamente funcional con una interfaz gráfica se facilita el uso del programa tanto a usuarios nóveles como a personas sin conocimientos de informática, como puede ocurrir en el caso de que el programa sea usado por investigadores médicos.

También se podría ampliar el programa para que soporte ficheros de entrada de otros programas, especialmente de Elvira, otro simulador cardiaco, puesto que se trata del simulador cardiaco que el laboratorio de investigación cardiaca de la UV usaba anteriormente.

Finalmente, se podría haber implementado algún método de escritura a ficheros más eficiente. El programa genera los ficheros insertando la información necesaria de forma estándar, con el operador de inserción. Sin embargo, las instrucciones de entrada y salida a ficheros son costosos debido, en parte, a la escritura en disco. Para reducir el número de llamadas al disco duro se podría haber implementado un método que usara un buffer para ir almacenando la información y solo escribir a disco cuando este estuviera lleno. O, otra opción sería, hacer que el programa realizara la escritura a disco de forma paralela al resto del programa mediante programación concurrente.

10 Cierre del Proyecto

El proyecto ha presentado una desviación temporal de 19 días laborales respecto a la planificación original. La fecha de finalización original era el 4 de Octubre, mientras que el proyecto ha sido finalizado el 31 de Octubre.

Uno de los motivos que ha causado el retraso del proyecto fue haber planificado que se iba a trabajar en el proyecto a jornada completa (8 horas al día, 5 días a la semana). En la realidad el horario ha sido mucho más irregular, trabajando algún fin de semana, algunos días se ha trabajado más de 8 horas y otros varias menos.

Otra causa que ha demorado el fin del proyecto ha sido que se subestimo la cantidad de horas necesarias que se emplearon para familiarizarse con todas las herramientas y conocimientos necesarios para emprender el proyecto. Para realizar el proyecto el alumno ha tenido que aprender cómo funciona VTK, una librería muy compleja con muchísimas clases y métodos, CARP el simulador cardiaco también muy complejo que dificulto la realización de las pruebas en el marco temporal planificado e informarse en profundidad sobre el funcionamiento del corazón, algo sobre el que solo se poseía conocimientos muy básicos. Todo esto dificulto tanto el progreso del proyecto como conseguir realizar una estimación temporal precisa.

El coste del proyecto era en su mayoría el coste de la mano de obra, por lo que al coste final del proyecto se le han añadido las 152 horas laborales extras que ha supuesto realizar el proyecto. El coste final del proyecto es por lo tanto el siguiente.

2. PRECIO DE VENTA

2.A MARGEN

		SIN CONTINGENCIAS		CON CONTINGENCIAS	
CONCEPTO	**MARGEN**	**IMPORTE**	**V. MARGEN**	**IMPORTE**	**V. MARGEN**
PERSONAL	20.00%	12,559.97 €	2,511.99 €	13,564.77 €	2,712.95 €
AMORTIZACION HW/SW	20.00%	24.65 €	4.93 €	26.63 €	5.33 €
PORCENTAJE MARGEN PROYECTO	20.00%		**2,516.92 €**		**2,718.28 €**

2.B PRECIO DE VENTA

PRECIO DE VENTA, Euros, SIN ESCALACION INTERANUAL			**15,101.55 €**	**16,309.67 €**
PRECIO DE VENTA, Euros, CON IVA al ...	**21.00%**	**TOTAL**	**18,272.87 €**	**19,734.70 €**

Tabla 12. Coste final del proyecto.

11 Anexos

11.1 Anexo 1: Sintaxis de un fichero .elem en CARP

Suponiendo una malla de n elementos donde:

- T_i el tipo de figura geométrica del elemento i.
- $N_{i,j}$ indexa el nodo j del elemento i.
- m_i número de nodos del elemento i.
- r_i región del elemento i (opcional).

Formato	**Ejemplo**
n	`20000`
$T_0\ N_{0,0}\ N_{0,1} \ldots N_{0,m_{i-1}}\ [r_0]$	`Tt 10 15 123 45`
$T_1\ N_{1,0}\ N_{1,1} \ldots N_{1,m_{i-1}}\ [r_1]$	`Ln 5 20  1`
...	...
$T_{n-1}\ N_{n-1,0}\ N_{n-1,1} \ldots N_{n-1,m_{i-1}}\ [r_{n-1}]$	`Tr 9545 456 10`

11.2 Anexo 2: Sintaxis de un fichero .pkje en CARP

```
97 # Número de cables presentes
##################################################
Cable 0 # El cable 0 siempre es el primero
-1 -1 # El cable 0 no tiene padres
1 2 # Los cables 1 y 2 son sus hijos
4 # Número de nodos en este cable
75 # Tamaño relativo del cable (número de fibras paralelas
   # o área transversal
100.0 # resistencia de la unión gap (kOhm)
0.0006 # conductividad (Ohm-cm)
1100 -3400 2200 # Posición Nodo 1
1149 -3300 2178 # Posición Nodo 2
1184 -3201 2165 # Posición Nodo 3
1201 -3090 2154 # Posición Nodo 4
##################################################
Cable 1 # El cable 1 (hay que seguir el orden)
.
.
.
```

11.3 Anexo 3: Plantilla base del fichero de parámetros en CARP

```
#carp.petsc.pt +F parameter.par

# Parameter file
simID = MySim #directorio donde los resultados son escritos
dt = 10 #tamaño en us del paso del tiempo
tend = 1000 #tiempo total de la simulación
meshname = 1corazon #nombre base de los ficheros de modelos
num_stim = 0 #Número de estímulos eléctricos a aplicar

purkf = arbol1 #nombre base del fichero con la geometría de
#Purkinje
purkEleType = 1 #Tipo de elemento a usar. Cubic hermite (0)
#o lineal (1)
His_n = 1 #Número de activaciones del haz de His a intentar
His_curr = 300 #Corriente en uA/cm3 a inyectar en el haz de
#His
Kfile = Kfile

dump_im = 1 #Volcar las corrientes de membrana. Sí (1) o no
#(0)
vofile = electricidades #Fichero de imagen IGB con los
#valores de los voltajes
```

12 Bibliografía

[1] Aparato circulatorio. (n.d.). *Wikipedia*. Consultado 21 de febrero de 2018, de https://es.wikipedia.org/wiki/Aparato_circulatorio

[2] Pulmones. (n.d). *Wikipedia*. Consultado 21 de febrero de 2018 de https://es.wikipedia.org/wiki/Pulmones

[3] OpenStax, Anatomy & Physiology. OpenStax CNX. Consultado 23 de febrero de 2018, de https://cnx.org/contents/FPtK1zmh@8.108:Y5T_wVSC@4/Heart-Anatomy

[4] Mihail, F. (2014). *Atlas of Heart Anatomy and Development.* Londres: Springer.

[5] Hall, J. (2015). *Guyton and Hall textbook of medical physiology.* Filadelfia: Elsevier

[6] Delille, J., Hernigou, A., Sene, V. et al. Eur Radiol (1999) 9: 1183. https://doi.org/10.1007/s003300050814

[7] Hoit, B. (2017). Anatomy and Physiology of the Pericardium. *Cardiology Clinics, 35*(11), 481-490. https://doi.org/10.1016/j.ccl.2017.07.002.

[8] The Heart. (n.d). En *Lumen Learning.* Consultado 9 de abril de 2018, de https://courses.lumenlearning.com/boundless-ap/chapter/the-heart/

[9] Cardiac muscle cell. (n.d). En *Wikipedia.* Consultado 11 de abril de 2018, de https://en.wikipedia.org/wiki/Cardiac_muscle_cell

[10] Cardiac muscle. (n.d). *Wikipedia.* Consultado 29 de abril de 2018, de https://en.wikipedia.org/wiki/Cardiac_muscle

[11] Membrane potential. (n.d). *Wikipedia.* Consultado 24 de abril de 2018, de https://en.wikipedia.org/wiki/Membrane_potential

[12] Action potential. (n.d). *Wikipedia.* Consultado 11 de abril de 2018, de https://en.wikipedia.org/wiki/Action_potential

[13] OpenStax, Anatomy & Physiology. OpenStax CNX. Consultado 11 de abril de 2018, de https://cnx.org/contents/FPtK1zmh@9.1:MCgS6S0t@4/Cardiac-Muscle-and-Electrical-Activity

[14] Conorary circulation. (n.d). *Wikipedia*. Consultado 9 de abril de 2018, en https://en.wikipedia.org/wiki/Coronary_circulation

[15] Conorary Arteries. (n.d). *Universidad de Minesota*. Consultado 9 de abril de 2018, de http://www.vhlab.umn.edu/atlas/coronary-arteries/index.shtml

[16] Miocardial infarction. (n.d). *Wikipedia.* Consultado 12 de abril de 2018, de https://en.wikipedia.org/wiki/Myocardial_infarction

[17] Muscle. (n.d). *Wikipedia.* Consultado 26 de abril de 2018, de https://en.wikipedia.org/wiki/Muscle

[18] Electrolyte. (n.d). *Wikipedia.* Consultado 25 de abril de 2018, de https://en.wikipedia.org/wiki/Electrolyte

[19] Membrane potential. (n.d). *Wikipedia.* Consultado 26 de abril de 2018, de https://en.wikipedia.org/wiki/Membrane_potential

[20] Ion transporter. (n.d). *Wikipedia.* Consultado 29 de abril de 2018, de https://en.wikipedia.org/wiki/Ion_transporter

[21] Gating (electrophysiology). (n.d). *Wikipedia.* Consultado 26 de abril de 2018, de https://en.wikipedia.org/wiki/Gating_(electrophysiology)

[22] Cardiac action potential. (n.d). *Wikipedia.* Consultado 26 de abril de 2018, de https://en.wikipedia.org/wiki/Cardiac_action_potential

[23] T-type calcium channel. (n.d). *Wikipedia.* Consultado 26 de abril de 2018, de https://en.wikipedia.org/wiki/T-type_calcium_channel

[24] Santana, L., Cheng, E., & Lederer, W. (2010). How does the shape of the cardiac action potential control calcium signaling and contraction in the heart?. *Journal of Molecular and Cellular Cardiology,* 49(6), 901-903. https://doi.org/10.1016/j.yjmcc.2010.09.005.

[25] Shih, H. (1994). Anatomy of the Action Potential in the Heart. *Texas Heart Institute Journal,* 21(1), 30-41. Disponible en https://www.ncbi.nlm.nih.gov/pmc/articles/PMC325129/pdf/thij00036-0042.pdf

[26] Klabunde, R. Sinoatrial Node Action Potentials. (n.d). *Cardiovascular Physiology Concepts.* Consultado 30 de abril de 2018, de https://www.cvphysiology.com/Arrhythmias/A004

[27] Pacemaker current. (n.d). *Wikipedia.* Consultado 30 de abril de 2018, de https://en.wikipedia.org/wiki/Pacemaker_current

[28] HCN channel. (n.d). *Wikipedia.* Consultado 30 de abril de 2018, de https://en.wikipedia.org/wiki/HCN_channel

[29] Dobrzynski, H., Boyett, M., & Andersor, H. (2017). New Insights Into Pacemaker Activity. *Circulation*, 115(14), 1921-1932. Disponible en https://www.ahajournals.org/doi/full/10.1161/circulationaha.106.616011

[30] Cardiac pacemaker. (n.d). *Wikipedia.* Consultado 01 de mayo de 2018, de https://en.wikipedia.org/wiki/Cardiac_pacemaker

[31] Sinoatrial node. (n.d). *Wikipedia.* Consultado 02 de mayo de 2018, de https://en.wikipedia.org/wiki/Sinoatrial_node

[32] Kafer, CJ. (1991). Internodal pathways in the human atria: a model study. *Computer Biomed Research*, 24(6), 549-563. Disponible en https://www.ncbi.nlm.nih.gov/pubmed/1769231

[33] Electrocardiography. (n.d). *Wikipedia.* Consultado 13 de mayo de 2018, de https://en.wikipedia.org/wiki/Electrocardiography

[34] Magnetic resonance imaging. (n.d). *Wikipedia.* Consultado 13 de mayo de 2018, de https://en.wikipedia.org/wiki/Magnetic_resonance_imaging

[35] Okajima, M., Fujino, T., Kobayashi, T., & Yamada, K. (1968). Computer Simulation of the Propagation Process in Excitation of the Ventricles. *Circularion Research*, 23(2), 203-211. Disponible en http://circres.ahajournals.org/content/23/2/203

[36] He, B. (2004). *Modeling and Imaging of Bioelectrical Activity Principles and Applications.* Estados Unidos: Springer.

[37] Lorange, M., & Gulrajani, R. (1993). A computer heart model incorporating anisotropic propagation. *Journal of Electrocardiology.* 26(4), 245-261. Disponible en https://www.jecgonline.com/article/0022-0736(93)90047-H/fulltext

[38] The Visible Human Project. (2003). U.S National Library of Medicine. Consultado 13 de mayo de 2018, de https://www.nlm.nih.gov/research/visible/visible_human.html

[39] Balasubramaniam, C., Gopakumaran, B., & Jagadeesh JM. (1997). Simulation of cardiac conduction system in distributed computer environment. *Biomedical Science Instrumentation*, 33, 13-18. Disponible en https://www.ncbi.nlm.nih.gov/pubmed/9731328

[40] Kauppinen, P., Hyttinen, J., Laarne P., & Malmivuo, J. (1999). A software implementation for detailed volume conductor modelling in electrophysiology using finite difference method. *Computer Methods and Programs in Biomedicine,* 58(2). Disponible en https://www.sciencedirect.com/science/article/pii/S0169260798000844?via%3Dihub

[41] Hren, R., Stroink, G. & Horáček, B.M. Med. Biol. Eng. Comput. (1998) 36: 145. https://doi.org/10.1007/BF02510736

[42] Toshimori, H., Toshimori, K., Oura, C., Matsuo, H., & Matsukura, S. (1988). Immunohistochemical identification of Purkinje fibers and transitional cells in a terminal portion of the impulse-conducting system of porcine heart. *Cell and Tissue Research,* 253(1), 47-53. Disponible en https://www.ncbi.nlm.nih.gov/pubmed/2843286

[43] Ansari, A., Ho, S., & Anderson, R. (1999). Distribution of the Purkinje Fibres in the Sheep Heart. *The anatomical record,* 245, 92-37. Disponible en https://onlinelibrary.wiley.com/doi/epdf/10.1002/%28SICI%291097-0185%2819990101%29254%3A1%3C92%3A%3AAID-AR12%3E3.0.CO%3B2-3

[44] Durrer, D., Dam, R., Freud, G., Janse, M., Meijler, F., & Arzbaecher, R. (1969). Total Excitation of the Isolated Human Heart. *Circulation,* 41(6), 899-912. Disponible en https://www.ahajournals.org/doi/abs/10.1161/circ.41.6.899#

[45] M. Aoki, Y. Okamoto, T. Musha and K. Harumi, "Three-Dimensional Simulation of the Ventricular Depolarization and Repolarization Processes and Body Surface Potentials: Nornal Heart and Bundle Branch Block," in IEEE Transactions on Biomedical Engineering, vol. BME-34, no. 6, pp. 454-462, Junio 1987. doi: 10.1109/TBME.1987.326079 Consultado en https://ieeexplore.ieee.org/document/4122568/

[46] Simelius, K., Nenonen, J., & Horáček, M. (2001). Modeling Cardiac Ventricular Activation. *International Journal of Bioelectromagnetism,* 3(2), 51-58. Disponible en: http://lib.tkk.fi/Diss/2004/isbn9512270285/article6.pdf

[47] Johnson, C. (2000). "Numerical Methods for Bioelectric Problems". *The Biomedical Engineering Handbook: Second Edition.* Boca Ratón: CRC Press LLC.

[48] Kauppinen, P., Hyttinen, J., Laarne, P., & Malmivuo, J. (1999). A software implementation for detailed volume conductor modelling in electrophysiology using finite difference method. *Computer Methods and Programs in Biomedicine,* 58(2), 191-203. Disponible en https://www.sciencedirect.com/science/article/pii/S0169260798000844?via%3Dihub

[49] [Schuster Engineering]. (2017, Marzo 30). *FEA 01: What is FEA?* [Fichero de video]. Obtenido de https://www.youtube.com/watch?v=sSaUHDQf204

[50] The Origins of VTK. (n.d). Kitware. Consultado 15 de mayo de 2018, de https://www.vtk.org/overview/

[51] Platform Agnostic. (n.d). Kitware. Consultado 15 de mayo de 2018, de https://www.vtk.org/features-platform-agnostic/

[52] Language Agnostic. (n.d). Kitware. Consultado 15 de mayo de 2018, de https://www.vtk.org/features-language-agnostic/

[53] The Origins of VTK. (n.d). Consultado 15 de mayo de 2018, de https://www.openhub.net/p/vtk

[54] Participants. (n.d). Kitware. Consultado 15 de mayo de 2018, de https://www.vtk.org/participants/

[55] Visualization Toolkit (VTK) Tutorial. (2004). Bell, J. Consultado 26 de junio de 2018, de https://www.cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html

[56] Avila, L., et al. (2010). *The VTK User's Guide*. Columbia: Kitware, Inc. Disponible en https://www.kitware.com/products/books/VTKUsersGuide.pdf

[57] Cardiac Arrhythmia Research Package. (2007). Consultado 06 de junio del 2018, de https://carp.medunigraz.at/

[58] Bidomain model. (n.d). *Wikipedia*. Consultado 06 de junio de 2018, de https://es.wikipedia.org/wiki/Bidomain_model

[59] Products and Services. (n.d). Consultado 06 de junio de 2018, de https://research.cardiosolv.com/services/

[60] TARANTULA. (2007). Consultado 06 de junio de 2018, de https://carp.medunigraz.at/04_meshGeneration/04_tarantula.htm

[61] Cross-Platform. (n.d). Consultado el 06 de junio de 2018, de https://www.visual-paradigm.com/VPGallery/crossplatform/index.html

[62] Guia del Mercado Laboral 2018. (2018). HAYS. Consultado 13 de junio de 2018, de http://guiasalarial.hays.es/trabajador/home

[63] Reglamento general de protección de datos 2016 (UE) (España). Disponible en http://boe.es/buscar/doc.php?id=DOUE-L-2016-80807